UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

Mobiltat de l'entorn i arquitectura del projecte SECURity at the network EDge

Treball de final de grau

Autor:
Ferran Pérez Gutierrez

Director:
René Serral Gracià

17 d'abril de 2017

Grau en Enginyeria Informàtica
Enginyeria de computadors

Resum

En el marc del projecte SECURED (SECURity at the network EDge),neix la necessitat d'implementar els mecanismes de mobilitat de l'entorn de l'usuari en l'arquitectura de confiança que ofereix SECURED, de forma transparent a l'usuari. La planificació ha estat acurada amb les necessitats del projecte i no s'han desviat gaire de la realitat. Els costos han estan ajustats dins de la partida pressupostaria de l'equip de la UPC. Es un projecte amb un valors de producció sostenible pel medi ambient, social i economic, segons la matriu de "la economía del bien común"[16], una proposta generada des de la FIB. Finalment la solució proposada s'ha provat en el centre d'ensenyament S'Arenal, del municipi Platja de Palma. La prova pilot ha sigut un èxit, demostrant així que la solució és exportable a qualsevol indret.

Resumen

En el marco del proyecto SECURED (SECURity at the network EDge), nace la necesidad de implementar los mecanismos de mobilidad del entorno del usuari en la aqrquitectura de confianza que ofreze SECURED, de forma transparente al usuario. La planificacion ha estado acurada con las necesidades del proyecto i se ha deviado mucho de la realidad. Los costes han sido ajustados dentro de la partida presupostaria del equipo de la UPC. Es un proyecto con unos valores de producción sostenible para el medio ambiente, social i enomico, según la matriz de l·la economía del bien común"[16], una propuesta generada des de la FIB. Finalmente la solución propuesta se ha probado en el centro de enseñanza S'Arenal, del municipio Platja de Palma. La prueba ha sido un exito, demostrando de así que la solución es exportable a cualquier sitio.

Abstract

SECURED (security at the network edge) project fullfilled the need of implementing the user's mobility mechanisms in the trusted architecture that SECURED offers, in a transparent way to the user. The planning has been careful to the needs of the project and has not deviated much from reality. The costs are adjusted within the budget items of equipment at the UPC. It is a project with sustainable production values for the environment, social and economic, as the matrix of l·la economía del bien común"[16] generated as a proposal from the FIB. Finally, the proposed solution has been tested in the training center of El Arenal municipality of Playa de Palma. The pilot was a success, showing that the solution is exportable anywhere.

Agraïments

Agraïments per a **Diego Montero** pe la seva ajuda incondicional, gracies això he pogut avançar en el moment crítics del projecte. També els meus agraïments a **Alícia Vila** pel temps que ha dedicat, amb un servidor, a posar en comú les parts de cadacú ha desenvolupat del projecte. I per ultim el meus agraïments a **René Serral Gracià** per l'oportunitat que m'ha brindat, per poder realitzar el TFG, dins d'un gran projecte com ha sigut SECURED, a més a més de la paciència que ha tingut amb mi.

Índex

Índex de taules

Índex de figures

Índex de codi font

Glossari

Antiphising El programari anti-phishing consisteix en programes informàtics que intenten identificar el contingut de suplantació d'identitat continguda en pàgines web i correu electrònic o bloquejar als usuaris ser enganyat. Sovint s'integra amb els navegadors web i clients de correu electrònic com una barra d'eines que mostra el nom de domini real de la pàgina web que l'espectador es troba de visita, en un intent d'evitar que els llocs web fraudulents es facin passar per altres llocs web legítims.

. 20

flag En la programació, un flag és una seqüència de bits o bits predefinit que conté un valor binari.. 30, 33

gunicorn Gunicorn 'Green Unicorn' és un servidor HTTP de Python WSGI per a UNIX. És un treballador model pre-fork. El servidor Gunicorn és àmpliament compatible amb diversos frameworks web, simplement implementat, baix consum de recursos del servidor, i bastant ràpida.. 22

OpenFlow OpenFlow és la primera interfície de comunicacions estàndard definit entre les capes de control i la transmissió d'una arquitectura SDN.. 29

outlier En estadística, una observació atípica o dada atípica o és un valor que difereix tan àmpliament de la resta de dades que podem pensar que s'ha comès un error.. 39

plugin Un connector (en anglès plugin, de plug-in: "endollar"), també conegut com a extensió (en anglès addin, add-in, addon o add-on) és una aplicació informàtica que interacciona amb una altra aplicació per aportar-li una funció o utilitat addicional, generalment molt específica, com per exemple servir com a controlador en una aplicació, per a fer així funcionar un dispositiu en un altre programa. 27

timestamps timestamping es l'ús d'una marca de temps electrònic per proporcionar un ordre temporal entre un conjunt d'esdeveniments.. 12, 38

VPN Una xarxa privada virtual, XPV o VPN (de les inicials de virtual private network) és una tecnologia de xarxa que permet una extensió de la xarxa local sobre una xarxa pública o no controlada. v, vi, 4, 7, 20, 27, 39

Capítol 1

Introducció

1.1 Descripció del projecte

Aquest treball final de grau, dins de la modalitat A, forma part d'un projecte major, finançat per l'Unió Europea amb participació de prestigioses universitats europees i empreses de telecomunicacions. El projecte es diu SECURED (SECURity at the network EDge), que ha durat tres anys de desenvolupament

Un dels grans contribució del projecte es la seguretat informàtica. Per això proposa una arquitectura innovadora per aconseguir protecció contra les amenaces d'Internet mitjançant l'encaminament del tràfic a través d'una sèrie de màquines on cadascuna s'encarrega de filtrar-lo seguint els criteris que l'usuari ha configurat.

Aquesta arquitectura crea un entorn d'execució de confiança i virtualitzat que permet diferents actors (per exemple, els usuaris individuals, administradors de TIC corporatius) per instal·lar sota demanda i executar diverses aplicacions al dispositiu de vora de la xarxa, la descàrrega dels dispositius mòbils, el que garanteix el compliment de facilitat d'ús específic i independent del dispositiu polítiques de seguretat i protecció uniforme a través de diferents dispositius i xarxes personals.[1]

Aquest projecte té per objectiu implementar els mecanismes de mobilitat de l'entorn de l'usuari en l'arquitectura de confiança que ofereix SECURED, de forma transparent a l'usuari. La implementació d'aquesta nova tecnologia es realitzarà de forma progressiva. L'arquitectura proposada es validarà inicialment en un banc de proves controlat i finalment en un entorn real.

1.2 Actors implicats

En projecte SECURED intervenen un total de 7 socis, a continuació podem veure la taula amb el nom del soci, el país d'origen i el rol que desenvolupa cadascú dins del projecte.[2]

1.2.1 Lider d'equip

Cada soci te un líder d'equip que gestiona els recursos, la càrrega de feina, marca els objectius a curt termini i organitza les reunions. En René Serral Gracià és el líder del

Inicials	Nom complert	País	Rol
POLITO	Politecnico di Torino	Itàlia	coordinador
HPLB	Hewlett-Packard LTD	UK	soci
PTEL	Primetel PLC	Xipre	soci
TID	Telefónica Investigación y Desarrollo SA	Espanya	soci
UNICRI	United Nations Interregional Crime and justice Research Institute		soci
UPC	Universitat Politècnica de Catalunya	Espanya	soci
VTT	VTT Technical Research Centre of Finland	Finland	soci

Taula 1.1: Consorci

nostre equip i el director del meu treball de final de grau.

1.2.2 Equip UPC

El rol que té la UPC en el projecte es el següent:

- Definició dels requisits per a l'arquitectura, així com el seu disseny i avaluació.
- Definició de l'API, ontologies i les polítiques de resums tant les sol·licituds de seguretat i les configuracions requerides en els diferents nivells, amb la participació tant dels dispositius físics i virtuals.
- Orquestració i desenvolupament per atendre els usuaris mòbils.
- La creació de prototips i la integració.

D'aquest rols els que m'afecten directament són els d'orquestració i creació de prototips (migració) i la integració.

1.2.3 Altres socis

Ara explicaré breument el paper que desenvolupen altres socis del projecte SECURED:

1. POLITO
    - Coordinador del projecte SECURED.
    - Desenvolupament i recerca en les polítiques de seguretat.
    - Desenvolupament i recerca en programari i aplicacions per al dispositiu de vora de la xarxa, basat en la seva experiència en la velocitat de processament flexible i alt de tràfic de xarxa, els mètodes formals de verificació de codi de seguretat, i el recent treball d'exploració sobre la migració d'aplicacions de xarxa personals des del host fins a la vora de la xarxa.
2. HPLB
    - Realitzar la definició general de l'arquitectura tècnica per assegurar que serà comercialment rellevant i realista.

- Conduir el desenvolupament i la implementació de la xarxa fonamental arquitectura de dispositiu de vora.

3. PTEL

   - Les activitats de desenvolupament i recerca en el disseny i l'avaluació de l'arquitectura de SECURED, contribuint amb els requisits de la xarxa des de la perspectiva d'una operadora.
   - Recollida d'informació i retroalimentació dels usuaris finals.
   - Validació tècnica de SECURED.
   - La difusió dels resultats del projecte.

4. TID

   - Contribució a la definició de les necessitats i opcions d'arquitectura per a escenaris d'operador de xarxa.
   - Contribuir al disseny del servei de seguretat, davant escenaris heterogenis barreja equips de xarxa de llegat i nous nodes de vora, que incorporen també les normes de seguretat de l'operador i normes internes.
   - Proporcionar entorns de prova el real HW / SW en ús dins de Telefònica, i simular diferents tipus de mobilitat de trànsit.
   - Participen activament en l'estandardització a través dels òrgans de TID. Fan l'anàlisi, avaluació i validació de la viabilitat de propostes en l'entorn més ampli d'Internet.

5. UNICRI

   - Ajudant en els requeriments de pagaments de part dels interessats, amb especial atenció a tres àrees principals de risc: en línia de protecció de la infància, la lluita contra la delinqüència informàtica, el suport continu dels usuaris d'Internet contra els règims autoritaris.
   - L'execució del procés d'avaluació amb les parts interessades.
   - Contribuint en l'especificació de les polítiques sobre l'equilibri de la lluita contra la delinqüència informàtica i la privacitat, la protecció dels nens en línia i donar suport a la xarxa contra la censura i la vigilància.
   - Avaluació de la capacitat d'ús de la interfície d'usuari per a l'especificació de la política.

6. VTT

   - La investigació en les àrees de detecció d'intrusos i de forma anònima.
   - Que porta tota la feina d'integració.
   - Organitzar un laboratori de la integració.
   - Difusió i explotació a través de xarxes nacionals i internacionals VTT.

1.2.4 Beneficiari

Els beneficiaris directes del projectes son les pròpies empreses de telecomunicacions que son socis del projecte. Indirectament, les universitats implicades podràn fer us de part del projecte en noves vies de recerca. A més a més a el projecte un cop acabat es farà public i es tindrà accés a ell.

1.3 Estat de l'art

Aquest projecte pretén crear un producte que ofereixi una garantia de seguretat i portabilitat a qualsevol entorn. Per tant, gran part de l'arquitectura que es planteja és fer us de recursos de virtualització i de polítiques d'enrutament basades en L'estàndard Open-FLow [3]

1.3.1 Open-Flow

Open-Flow és un estàndard obert que permet als investigadors per executar els protocols experimentals en les xarxes de campus que fem servir cada dia. Open-Flow s'afegeix com una característica de commutadors Ethernet comercials, routers i punts d'accés sense fil, i proporciona un estàndard per permetre als investigadors a realitzar experiments, sense necessitat de venedors per exposar el funcionament intern dels seus dispositius de xarxa. Open-Flow actualment està sent implementat pels principals proveïdors.'

1.3.2 Virtualització

La majoria de les màquines gestió d'usuaris i control de l'entorn, son maquines virtuals fent servir kvm, una solució basada en el nucli Virtual Machine o KVM. És una solució per implementar virtualització completa amb Linux, formada per un mòdul del nucli (amb el nom kvm.ko) i eines en l'espai d'usuari, essent íntegrament programari lliure. El Component KVM Per El nucli està inclòs en Linux des de la versió 2.6.20.[**?**] KVM permet executar Màquines virtuals utilitzant imatges de discs durs que contenen Sistemes Operatius per Modificar. Cada màquina virtual té el seu propi hardware virtualitzat: discs durs, targetes gràfiques, etc.

A més a més farem us de Qemu, és un emulador de terminal de processadors per programari, que permet a un usuari simular un sistema complet dintre d'un altre. És programari lliure i és escrit per Fabrice Bellard. QEMU és un hypervisor i és similar a altres projectes com Bochs, VMware Worksation i PearPc.[4]

1.3.3 VPN

La connectivitat entre l'usuari i l'entorn de SECURED es realitzara per un túnel (VPN). Per tant es farà servir StrongSwan. És una implementació completa d'Internet Protocol Security (IPsec) per a Linux 2.6, 3.xi 4.x superior. L'objectiu del projecte és de tenir mecanismes d'autenticació forts mitjançant certificats de clau pública X.509 i emmagatzematge assegurança opcional de claus en targetes intel·ligents a través d'una interfície estandarditzada PKCS. [5]

S'utilizarà l'extensió MOBIKE[18] IKEv2 (RFC 4555), que permet canviar el final del túnel generat en la xarxa. I el plugin de mysql per suportar múltiples usuaris connectats.

En el capítol 6 s'aprofundirà amb més detall de com s'ha implementat.

1.3.4 Opendaylight

OpenDaylight [7] , es una plataforma SDN (Software-Defined Networking) de codi obert actual. Serà una solució temporal utilitzada inicialment en el projecte per orquestrar una migració d'una maquina virtual i per posar a punt el banc de proves.

1.4 Estructuració de document

A continuació tenim un breu resum de com esta estructurat el document.

Abast: Capítol on es formula el problema. Posteriorment s'expliquen els objectius per solucionar-lo, l'abast del projecte, riscos i possibles solucions que poden sorgir en el transcurs del projecte i el mètode de treball que s'utilitzarà.

Planificació: Capítol on es defineixen les tasques que s'han realitzat en el desenvolupament del projecte i el període dedicat.

Arquitectura: Capítol on es farà un resum de l'arquitectura del projecte SECURED, funcionalitats prèvies a la implementació i quin era l'escenari abans d'iniciar la implementació de la mobilitat.

Implementació: Capítol on es parlarà amb profunditat tècnica de totes les implementacions realitzades en el projecte.

Jocs de proves: Capítol on s'exposaran tots els jocs de proves que es realitzaran, quines dades es recolliran, en quines condicions i les característiques de l'entorn de proves on es faran.

Resultats: Capítol on mostraré les gràfiques obtingudes dels jocs de proves explicats l'apartat anterior.

Pressupost: Identificació i estimació de costos.

Sostenibilitat: Capítol que s'explicarà amb detall les avantatges mediambientals, socials i econòmiques que aportaria el projecte, un cop finalitzat, aplicat en un situació real.

Conclusions: Capítol final on es mostrarà fins a on s'ha arribat del projecte i futurs desenvolupaments.

Capítol 2

Abast

Seguidament s'explicaran el requeriments que ha de complir el projecte per justificar la inversió de temps, inversió econòmica i esforç per assolir els objectius amb èxit.

2.1 Formulació del problema

Protecció dels dispositius mòbils d'amenaces d'Internet s'aconsegueix normalment mitjançant la install·lació de les eines apropiades (per exemple, antivírics, tallafocs personal, control parental) en cada dispositiu.

Això, però, planteja diversos problemes, normalment es requereix un accés privilegiat al dispositiu, i molts cops les eines de protecció apropiades no estan disponibles per a totes les plataformes.

Un altre problema observat es el programari que ofereix un servei de seguretat. Normalment es especific en cada dispositiu. Amb un equip informàtics de sobretaula, inclòs portàtils, normalment el programari pot cobrir les necessitats, sempre i quan la connectivitat sigui en la mateix xarxa. Per tant només que canviem de xarxa, aquesta aplicació no sempre pot garantir la seguretat.
En els dispositius mòbils, observem que el programari normalment es limitat. A més a més aquestes eines poden consumir molts recursos, sobretot quan estan connectat a les xarxes de telefonia mòbil, actualment el cost en consum de dades es elevat. El dispositius mòbils constantment estan canviant d'una xarxa a una altre, independent que sigui el punt d'accés wifi de l'empresa, com la xarxa de l'operadora de telefònica que es tingui contractada. L'usuari no sap i desconeix que el canvi de xarxa sigui vulnerable pel seu dispositiu.

Un problema emergent, es protegir les generacions més joves d'amenaces o accés a contingut no apropiat per la seva edat. Els tutors legals no tenen cap eina per controlar i protegir aquest sector de la societat vulnerable.

Tot això, tradueix en la protecció insuficient pels usuaris amb una alta mobilitat, on els seus dispositius no pot garantir una connectivitat segura o controlada si és el cas d'un menor.

2.2 Objectiu

El projecte SECURED proposa una arquitectura innovadora per aconseguir protecció contra les amenaces d'Internet mitjançant la descàrrega i l'execució de les aplicacions de seguretat en un dispositiu programable pròxim al dispositiu de l'usuari, com un punt d'accés o un router de l'empresa.

L'arquitectura SECURED crea un ambient de confiança i virtualitzat que permet l'execució de diferents actors (per exemple, els usuaris individuals, administradors de TIC corporatius, proveïdors de la xarxa) per a instal·lar sota demanda i executar múltiples aplicacions de seguretat en el dispositiu que farà de punt d'accés a la xarxa, per protegir el tràfic d'un usuari específic. Aquesta solució redueix la càrrega i consum en els dispositius mòbils, el que garanteix el compliment de les polítiques d'usuari específics i independents del dispositiu de seguretat, i protecció uniforme a través de diferents dispositius i xarxes.

Mecanismes de transició entre xarxes es defineixen també per suportar els dispositius de xarxa tradicionals i implementar aquesta nova tecnologia de forma incremental. S'ha de garantir que tota arquitectura de l'entorn que sustenta la sessió de l'usuari es pugui replicar en el moment que l'usuari es connecti a un altre punt d'accés a la xarxa . Això implica migrar tot l'entorn virtualitzat, en calent. Totes les polítiques d'encaminament de les dades de l'usuari s'han de regenerar en el destí on es migrar tot l'entorn de l'usuari.

El projecte també serà capaç de fer complir les polítiques de seguretat sota demanda, no només quan l'empleat està connectat a la xarxa de l'empresa, sinó també quan aquest estigui en moviment.

2.3 Requeriments

2.3.1 Tecnologies

La connectivitat entre l'usuari i l'entorn que ofereix SECURED ha d'estar encriptat, per això es realitzara una connexió segura VPN.
Tecnologies de virtualització kvm i qemu han de permetre una migració rapida de l'entorn.
Les regles d'encaminament siguin fàcils de replicar d'un punt d'accés a un altre.
Per l'usuari ha de ser transparent i no ha de notar els canvis quan canvia de punt d'accés.

2.4 Abast

El resultat del projecte es oferir una solució còmode, transparent i segura de connectar-se a Internet. L'usuari no s'ha de fer càrrec de la seguretat, ja s'encarregara l'entorn que l'hi oferiria una operadora telefònica. I tindrà mobilitat total sense preocupar-se de perdre connectivitat, amb l'afegit de la seguretat que ofereix l'entorn.

2.5 Riscos i possibles solucions

Durant el projecte es poden donar algunes situacions problemàtiques. Per cada una d'aquestes s'intentarà donar solució apropiada:

2.5.1 Augment del temps en de depuració

Es un projecte de recerca, per tant es fa us de totes les eines més noves possibles per desenvolupar el projecte, això comporta un temps bastant elevat en depurar el correcte funcionament de tot l'ecosistema.
Solució: Plantejar el pitjor escenari i sempre treballar amb les ultimes versions estables de les tecnologies que farem servir i donar marge enorme en el moment de depurar abans de consumir el temps abans del següent pas.

2.5.2 Burocràcia

El projecte depèn del finançament de l'Unió Europea, cada soci te un pressupost diferent. Si en l'equip UPC es necessari obtenir nou maquinari que supera el pressupost, llavors estaríem parlant d'una limitació alhora de poder realitzar proves.

Solució: Ajustar-se al màxim amb les possibilitats que dona la partida pressupostaria per cada soci. Justificar obtenció de nou material fer una previsió de si aquest material serà útil fins al final del projecte. Si escau, reciclar els equips, que inicialment tenien una funció que ja no calen desenvolupar,per destinar-los a altres funcions que sí és necessari prioritzar per optimitzar recursos.

2.5.3 Maquines de prova

Tot l'entorn s'ha de provar en una serie de maquinari que ha d'estar en perfectes condicions, perquè si falla es genera un problema més a l'hora de desenvolupar.

Solució: Provar primer que el maquinari estigui en perfectes condicions per construir un entorn de proves. Amb un sistema operatiu net, sense cap configuració, excepte la que requereixi per realitzar les proves.

2.6 Identificació de lleis i regulacions

Pel que fa a les lleis i regulacions, aquest projecte no n'afegeix respecte el sistema actual, ja que el projecte es basa en construir un sistema que garanteixi la seguretat de les dades de l'usuari. Per tant, les lleis del tipus LOPD [6] es respecte perquè les dades sensibles dels usuaris mai son revelades i un cop l'usuari es desconnecta es destrueixen.

2.7 Mètode de treball

2.7.1 Desenvolupament semi-presencial

La major part del projecte es dur a terme a l'oficina, ja que les màquines estan en un entorn de prova que només és accessible des de les install·llacions de la UPC, per fer les proves d'integració s'ha de realitzar presencialment. El desenvolupament es pot realitzar a distància, amb l'inconvenient que la verificació de les noves implementacions s'han de realitzar presencialment sobre les maquines de prova.

2.7.2 Reunions de seguiment, de l'equip UPC

Es fan reunions de seguiment periòdiques, a nivell equip intern de la UPC, amb el lider d'equip amb la finalitat de comprovar l'evolució de la nostre part dins del marc projecte SECURED. En aquestes reunions s'exposaran tant els avanços com les possibles problemàtiques que puguin sorgir i es tracen accions per solucionar-ho per assolir els objectius.

2.7.3 Reunions de seguiment del projecte SECURED

Es fan reunions de seguiment mensuals, on es reuniran els líders de projecte de cada equip amb el director del projecte de amb la finalitat d'exposar la feina realitzada de cada part del projecte. EL avenços del que esta encara en desenvolupament. Finalment es detallen els pròxims objectius a realitzar amb de la següent reunió.

2.7.4 Proves i integració de l'entorn

1. L'entorn de proves que es muntarà per integrar la mobilitat de l'arquitectura SECURED, serà inicialment un entorn independent. Per tant es realitzarà en un entorn controlat i limitat.

2. Un cop el sistema sigui consistent es s'integrarà amb la resta dels components del projecte, com el PSAR[1] o la comunicació amb el client. En el capítol d'arquitectura de SECURED aprofundirem amb més detall.

3. Un cop funcioni amb els altres components etapa de depuració de tota L'arquitectura per optimitzar la migració de les maquines virtuals, l'entorn i la reconnexió de l'usuari en l'entorn en un nou punt d'accés.

4. I recoll·llectar dades per traçar una gràfica de temps en la migració.

5. Per últim queda fer una demostració final amb tot l'ecosistema, en un entorn real. Pot ser un institut, universitat, etc.

Capítol 3

Planificiació

3.1 Planificació temporal

En aquesta secció s'explicaran breument les tasques que s'han dut a terme durant el desenvolupament del projecte. Les tasques es descriuen en ordre cronològic i s'aprofundeixen degudament en el capítol d'implementació. Finalment podem veure gràficament les tasques amb el digrama de gantt, on es mostren els dies que s'han destinat a cada tasca.

3.1.1 Descripció de les tasques

Període d'adaptació a l'entorn del projecte SECURED

Les primeres setmanes bàsicament es fa un estudi de l'arquitectura del projecte SECURED[1], per saber en quin estat està el projecte i veure quines seran les pròximes tasques, com per exemple l'ús de maquines virtuals per treballar amb l'entorn. També quins són els llenguatges de programació que es faran servir. En aquest cas principalment serà Python per desenvolupar tot el procés que involucra la migració de l'entorn de l'usuari i la configuració de les polítiques d'encaminament de les dades.

Preparació de l'entorn de proves

Un cop assolit el coneixement de les eines i l'entorn, cal preparar el maquinari i el programari per realitzar el desenvolupament i posteriorment les proves de mobilitat del projecte SECURED. S'utilitzaran dos equips de format NUC [17] i un equip portàtil. Els NUCs es configuren com un punt d'accés. Aquests equips estan connectats entre ells, per simular una xarxa local, només visible entre els equips. Aquests equips faran la funció de NEDs. Per provar el funcionament del testbed es fa servir una maquina virtual com orquestrador per simular la migració d'una maquina virtual,amb opendaylight.[7]. Per fer la prova d'encaminament de dades es fa servir openvswitch.[8]

Configuració Strongswan

Aquesta tasca consisteix en instal·lar i configurar Strongswan. S'ha fet servir la versió 5.4.0 [5]. S'ha utilitzat el mòdul de mobike, per tant s'ha reconfigurat perquè escolti pel

port 4500, en comptes del port 500. Aquesta tasca esta explicada en més detall en el capítol de la implementació.

Integració de la mobilitat al projecte SECURED

La tasca més important, per tant la que ha cobert la major part del projecte, ha estat la implementació la nova nova classe del NED per realitzar la mobilitat de l'entorn de l'usuari (TVD), amb les seves PSAs i el PSC.
Finalment, en el NED destí es regeneren les polítiques d'encaminament openflow, gràcies a informació transmesa des del NED origen. I les maquines virtuals en destí. Gran part d'aquest bloc de tasques del projecte s'expliquen més detalladament en el capítol de la implementació, no obstant a continuació es farà un breu resum.

Primera versió del script de migració de les maquines virtuals Abans de res, es farà una versió més senzilla per provar el funcionament amb un script o classe simple amb Python. Es provarà en el entorn de proves. Un cop funcioni es passarà a la següent tasca.

Proves amb diferents tipus de migracions Es faran proves amb el script anteriorment creat fent càlculs del temps que tarda, fer copies incrementals del disc dur d'origen a destí, o fer una creació nova de disc dur. Migració temporal o persistent en destí. Convertir les maquines virtuals en "readonly", etc. Tot això servirà per veure quina es l'opció més optima per dur a terme en la mobilitat del projecte SECURED.

Adaptació del codi de migració de les VM Un cop es trobi l'opció amb menys cost de temps i recursos es crearà una classe nova a l'orquestrador del NED, anomenada TVDMigration, que s'encarregarà de fer la migració quan sigui el moment.

Programar la migració de la instància del TVD Un cop s'ha aconseguit que la migració de les maquines virtuals entre els NEDs, cal migrar i reinstanciar tot el TVD de l'usuari. Per realitzar les proves en el l'entorn de proves, s'improvisa un script en Python per simular l'aplicació d'usuari, per simular la comunicació amb el NED per realitzar la migració.

Integració de la mobilitat amb l'aplicació d'usuari L'aplicació d'usuari envia un missatge REST[9] json[12]. Un cop rebut s'executa la nova classe TVDMigration, que s'encarrega de migrar tot l'entorn (PSAs, PSC, variables de la instància de l'usuari TVD). En primer lloc comença a migrar-se les PSAs, un cop migrades, es migra el PSC i finalment s'enviarà un missatge REST json amb el TVD de l'usuari. Un cop acaben tots els events anteriors s'envia la resposta i tot seguit l'aplicació baixa el túnel strongswan en origen, seguidament es reconnecta al nou punt d'accés i finalment genera una nova connexió en destí.
Aquest punt s'explicarà en detall en el capítol 6.

Millora i paral·lelització del codi de mobilitat Un cop la mobilitat esta integrada amb l'aplicació, es millora la comunicació entre les dues parts, i es paral·lelitza el procés de migració de les PSAs i es crea un fluxe estable del proces de mobilitat, que serà el següent. S'inicia un thread de migració, dins d'aquest thread principal s'inicien n threads secundaris, un per cada PSAs, que realitzen paral·lelament la migració de les PSAs. Un cop que aquests threads secundaris assoleixin un 80% de la migració, envien un event al thread principal, aquest alhora s'inicia el thread secundari per realitzar la migració del PSC. En aquest punt el NED d'origen envia al NED destí, mitjançant un missatge REST json el TVD de l'usuari. Un cop acaben i realitzen el join tots els threads amb un event continua l'execució per enviar la resposta al l'aplicació del client.

Optimització de la comunicació entre l'aplicació i el NED Un cop aconseguit l'objectiu d'optimitzar el temps d'execució de la migració, cal millorar la comunicació de l'aplicació i el NED perquè en el cas d'haver múltiples usuaris funcioni a la perfecció. Aquesta tasca s'explica de forma més detallada en el capítol d'implementació.

Depurar el codi per la demostració de Mallorca El setembre del 2016 es fa una prova real a Mallorca, per tant es centra tots els recursos en mantenir estable tot l'entorn de mobilitat.
En els annexes hi ha un enllaç a la noticia referent a la prova pilot realitzada al S'Arenal, de la web del Diario de Mallorca

Recoll·llecció de dades per traçar un gràfica de temps en la migració

Es modifica el codi per afegir timestamps per obtenir, de la forma més precís, els temps en la migració de l'entorn. Aquest punt s'aprofundeix en el capítol de joc de proves i en el de resultats.

Depurar el codi per la publicació

Per ultim es neteja el codi de comentaris, línies de codi innecessàries, es fa la redacció de la documentació tècnica que es sol·licita per ser entregat al socis de POLITO per posar-ho de domini públic.

3.1.2 Diagrama de Gantt

Projecte	Data inici prevista	Dies treballats	Data final prevista
1. Període d'adaptació a l'entorn	1-jun.-15	29	30-jun.-15
2. Preparació del test-bed	1-jul.-15	30	31-jul.-15
3. Configuració Strongswan	1-sep.-15	29	30-sep.-15
4. Integració de la mobilitat a SECURED	1-oct.-15	365	30-sep.-16
4.1. Primera versió del script de migració	1-oct.-15	45	15-nov.-15
4.2. Proves amb diferents tipus de migracions	16-nov.-15	45	31-dic.-15
4.3. Adaptació del codi de migració de les VM	1-ene.-16	62	3-mar.-16
4.4. Programar la migració de la instància del TVD	1-mar.-16	31	1-abr.-16
4.5. Integració de la mobilitat amb l'app d'usuari	4-abr.-16	29	3-may.-16
4.6. Millora i paral·lelització del codi de mobilitat	1-may.-16	45	15-jun.-16
4.7. Optimització de la comunicació	15-jun.-16	46	31-jul.-16
4.8. Depurar el codi per la demostració de Mallorca	1-sep.-16	29	30-sep.-16
5. Millora del codi per suportar múltiples usuaris	3-oct.-16	12	15-oct.-16
6. Recol·lecció de dades	17-oct.-16	29	15-nov.-16
7. Depurar el codi per la publicació	15-nov.-16	46	31-dic.-16

Figura 3.1: Taula de les tasques, data d'inici, duració i data de finalització

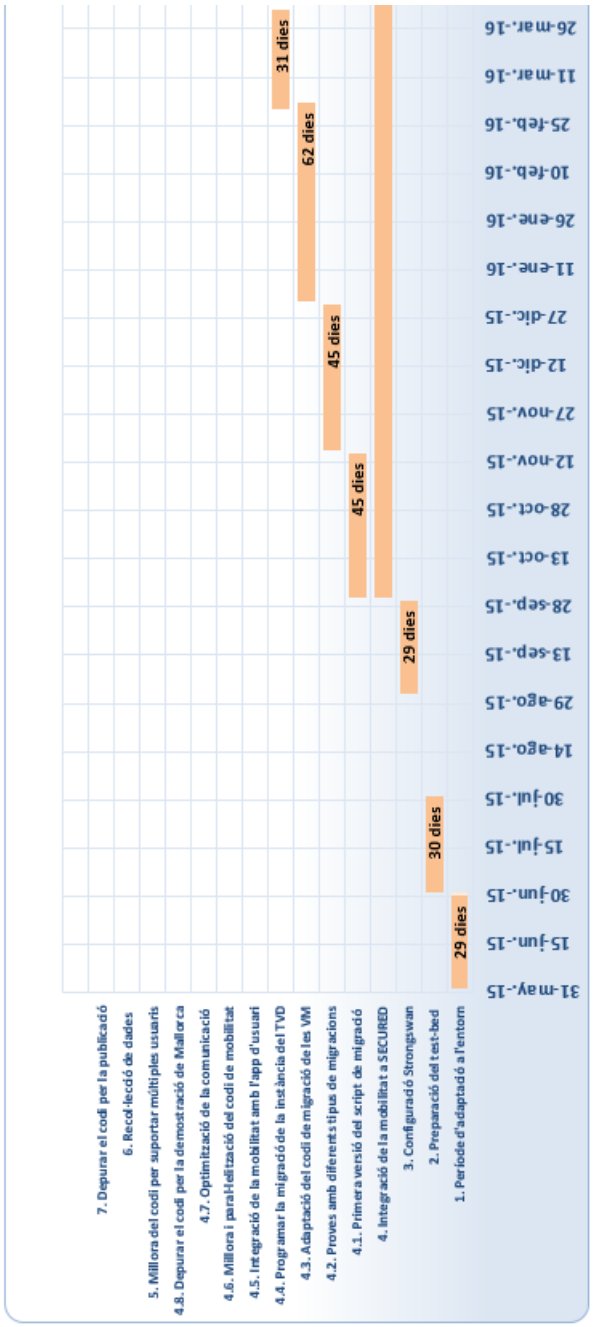


Figura 3.2: 1ª part del diagrama de gantt

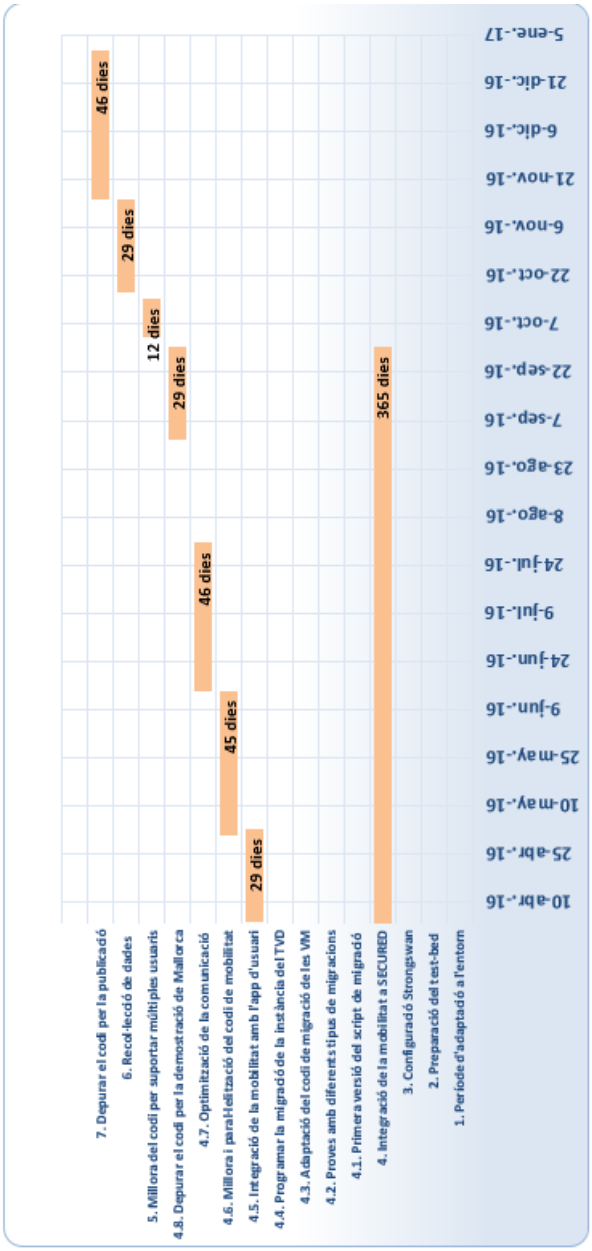


Figura 3.3: 2ª part del diagrama de gantt

Capítol 4

Pressupost

4.1 Pressupost

En aquest capitol es fa una identificació i estimació de costs de la part de mobilitat del projecte SECURED.

4.2 Identificació i estimació de costos

Al llarg del projecte es poden agrupar les diferent despeses en tres categories diferents:

- Equip de desenvolupament
- Maquinari
- Costos indirectes al projecte

4.2.1 Equip de desenvolupament

Les despeses que aquí es recullen, tenen relació amb el pagament de salaris als treballadors. Els membres de l'equip de desenvolupament que participa d'aquest projecte són:

- L'estudiant, assumirà el rol de desenvolupador d'una part de mobilitat del projecte SECURED, dedicant el total de la seva jornada laboral a aquest.
- *Assistent i desenvolupador*, es el rol que assumirà un treballador en assessorar a l'estudiant, en aspectes tècnics del projecte, quan així ho necessiti. També desenvoluparà parts necessàries per a mobilitat de SECURED.
- *Project leader*, responsable dins de l'equip UPC que gestiona el projecte en tot el seu desenvolupament i assessorar a l'estudiant quan així ho necessiti.

Per als anteriors participants, cal tenir en compte un seguit de consideracions:

1. Per a l'estudiant, queda pactat un contracte de recerca, destinat la seva totalitat amb el seu sou. En aquest cas són 10800 € a repartir entre els 18 mesos de durada del projecte.
2. En el cas de l'assistent, esta involucrat en el projecte SECURED, amb una remuneració de 1000 €

3. En el cas del *Project leader*, s'ha estimat que las seva feina equival a una remuneració mensual de45000 €.

Tenint en compte les anteriors consideracions, la taula de costos de personal esdevé de la següent manera:

Rol	**Concepte**	**Import**
Desenvolupador		
	Salari	10800€
Assistent		
	Salari	18000€
Project leader		
	Gestió del projecte	2500€
	Total	**57600€**

Taula 4.1: Taula de costos de personal

4.2.2 Maquinari

Corresponent a les eines de treball que es posen a disposició de l'estudiant per tal que pugui dur a terme la seva activitat dins de l'empresa.

El maquinari que s'ha posat a disposició de l'estudiant és:

- Estació de treball, amb un valor de compra de 900€
- Dos equips per les proves, en format NUC, amb un valor de compra de 500€/unitat

Al tractar-se de bens comprats per a l'ocasió, però que es faran servir en més projectes, no es pot imputar l'import complet, si no que s'aplicarà l'amortització pertinent als 18 mesos que dura el projecte.

La taula d'amortitzacions és la següent Tenint en compte la taula anterior, els imports

Tipus d'element	**Coeficient màxim**	**Període d'anys màxim**
Equips electrònics	20%	10

Taula 4.2: Taula d'amortitzacions (a partir de 1/6/2015)

imputables per cada element són:

- Cost imputable de l'estació de treball

$$\frac{900 * 20\%}{18 mesos} = 10€/mes$$

$$10€/mes * 18 mesos = \mathbf{180€}$$

- Cost imputable del NUCs

$$\frac{(2 * 500€/NUC) * 20\%}{18 mesos} = 11.11€/mes$$

$$11.11€/mes * 18 mesos = \mathbf{200€}$$

4.2.3 Costos indirectes

Dins del concepte de despeses indirectes s'hi inclou tot allò no relacionat directament amb el projecte, si no que s'apliquen a l'empresa en general.
Les despeses indirectes es poden classificar en:

- Despeses en concepte de llum, aigua, gas, un total de 200 € mensuals.

Per tant a la suma de costos del projecte, cal afegir un import de 200 € mensuals, en concepte de costos indirectes.

4.2.4 Taula de costos

A la següent taula es pot veure, a mode de resum, les despeses del projecte per als seus 18 mesos de durada:

Concepte		**Cost Mensual**	**Cost Total**
Personal			
	Desenvolupador	600 €	10800 €
	Assistent & dev.	1000 €	18000 €
	Project Leader	2500 €	45000 €
Maquinari			
	Estació de treball	10.0 €	180 €
	NUCs	11.11 €	200 €
Infraestructura			
	Despeses vàries	200 €	3600 €
Total			**61580 €**

Taula 4.3: Taula de costos del projecte

Capítol 5

Arquitectura

El projecte SECURED té com a objectiu proporcionar una arquitectura innovadora per a la protecció d'amenaces externes dels dispositius, mitjançant l'execució de les aplicacions de seguretat més comunes en un dispositiu programable en els punts d'accés a la xarxa. En aquest capítol descriurem detalladament l'arquitectura global els components que la formen, fent especial èmfasi en el , , PSAs i el de l'usuari i el NED que engloba tots aquest elements

En la Figura 5.1 observem la fotografia general del NED amb els components interns que seran descrits a continuació.

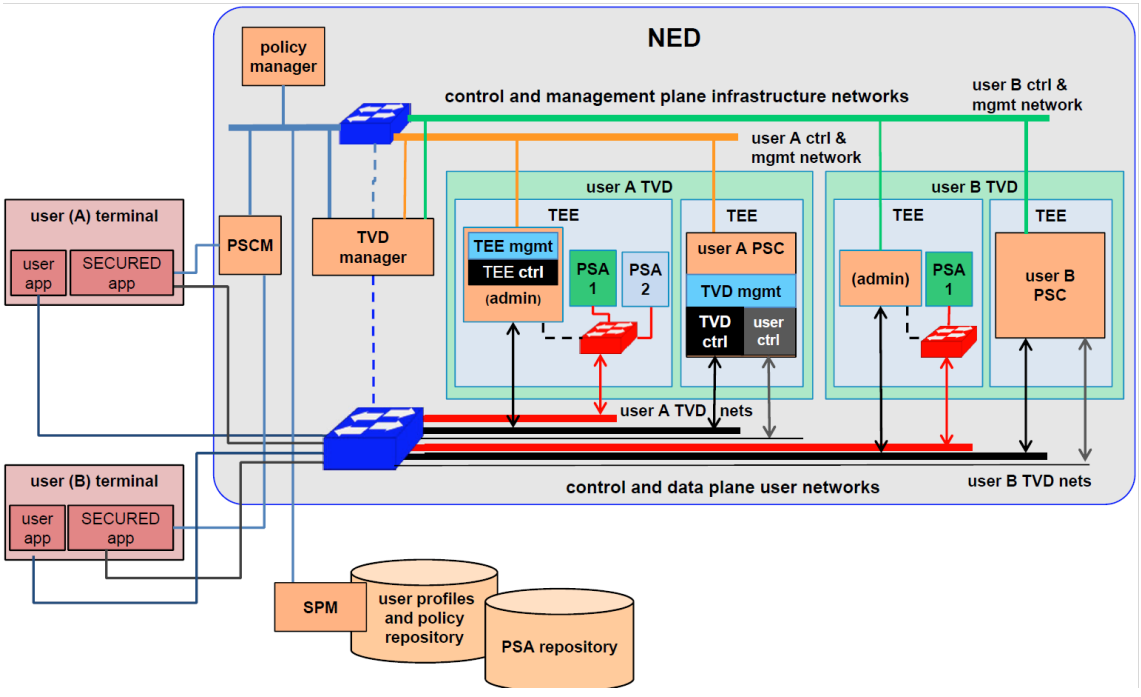


Figura 5.1: NED

5.1 Personal Security Application (PSA)

La Personal Security Application és l'entitat atòmica de l'arquitectura SECURED. És responsable de fer complir una o múltiples polítiques de seguretat dels usuaris. Cada PSA esta composada per una sèrie de controls de seguretat. Aquesta realitza una serie d'operacions primitives orientades al processament de la xarxa, com ara el filtratge, segmentació, càrrega i reensamblatge dels paquets.

Una política de seguretat complexa pot ser aplicada per un sol PSA o mitjançant la connexió de múltiples PSAs. L'encadenament podria estar entre els PSA que s'executen dins de la mateixa TEE o entre PSAs que s'executen en diferents TEEs però pertanyents a la mateixa TVD. Una sèrie de PSAs connectats correctament a l'aplicació d'un servei de seguretat implementa el concepte de feix de PSAs.

L'arquitectura PSA internament s'ha de dividir en dos plans: Control-gestió i dades.

1. **Control & Management plane**: és la part del programari que proporciona les interfícies cap al control i la part de gestió de la TEE per a la configuració, monitorització i peticions de senyalització des de i cap al PSC.

2. **Data plane**: és la part de la PSA que està a càrrec del processament en el trànsit d'entrada/sortida de línia, proporcionant interfície per a la capa de comunicació de l'entorn d'execució i permetent intercanvi de paquets entre els diferents anuncis de servei públic i també externament. La lògica interna del PSA es pot dividir en diferents mòduls.

5.1.1 Tipus de PSAs

Antiphising

LA PSA anti-phishing[10], com bé diu el seu nom, intenta identificar el contingut de suplantació d'identitat continguda en pàgines web i correu electrònic o bloquejar als usuaris ser enganyat.

VPN corporate

La PSA VPN corporate, et permet connectar a la xarxa de la teva empresa i sortir a Internet emulant que estas dins a la xarxa d'aquesta.

Control Parental

La PSA control parental, es el servei més potent pels pares per controlar els dispositius dels seus fills i evitar que entrin en llocs de contingut per adults i no desitjat.

Altres

Hi ha més tipus de PSAs, no obstant en el projecte de mobilitat es va treballar amb aquestes tres. Per tant només esmentem que hi han més tipus de funcionalitats.

5.2 TEE

Cada compartiment d'un usuari conté un entorn d'execució. Aquest ambient ha de ser prou segur i de confiança per separar les aplicacions concurrents sense interferir entre si. Es necessita un accés privilegiat per poder gestionar, configurar i iniciar les aplicacions. Les aplicacions es despleguen com veiem en la figura , en una àrea sense privilegis del TEE.

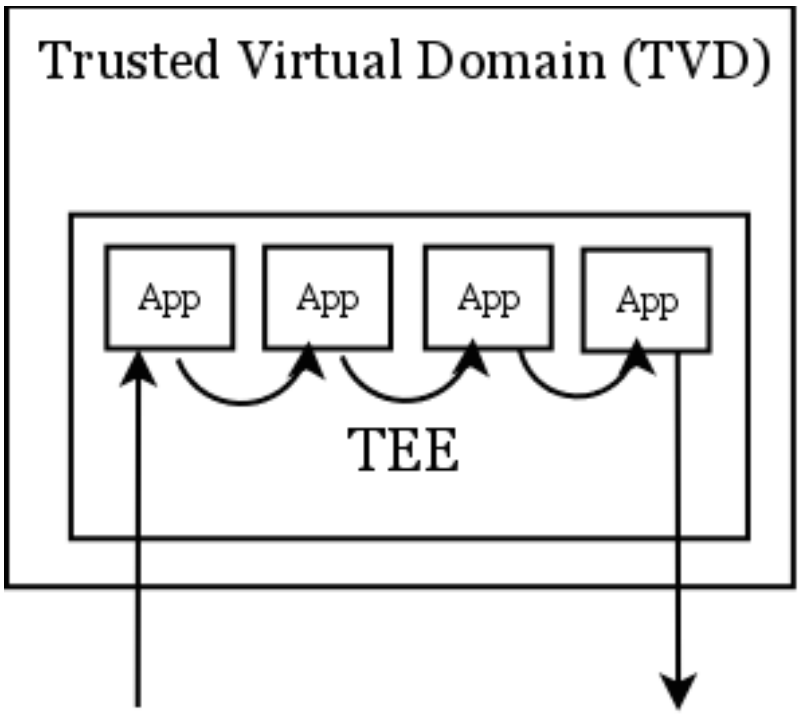


Figura 5.2: NED

5.3 Personal Secured Control (PSC)

El controlador de seguretat personal (PSC) és el component que controla i gestiona el TVD, els agents d'usuari, configuracions i interfícies de xarxa en nom d'un únic usuari. Cada usuari té la seva pròpia instància d'un PSC dins del seu TVD, que captura tot l'estat de l'usuari i el context de la seva sessió. Ja que el PSC té el privilegi de carregar aplicacions i fer complir les configuracions, per exigències del disseny esta aïllat de les aplicacions.Per tant totes les aplicacions relacionades amb l'usuari s'han de comunicar al seu PSC a través d'una interfície de Control i Administració restringit.

No obstant això, els canvis en la TVD global (com la creació d'un nou TEE) requereixen l'enviament d'una sol·licitud a l'orquestrador, que és una entitat privilegiada extern fora de la TVD.

Funcions principals:

1. Emmagatzema el gràfic de servei abstracte (PSA) i les interconnexions de l'usuari;
2. determina la topologia de TVD (realització del gràfic de servei) dels requisits de PSA (això pot canviar depenent dels recursos NED);

3. envia la topologia TVD a la TVDM (aquest últim conté tota la topologia TVDs i com 1 TVD està connectat a un altre);

4. supervisa l'estat de la TVD (detectar per exemple si el TEE s'ha estavellat i necessita reiniciar);

5. PSC rep els manifestos de tots els anuncis de servei públic i s'assigna a cada PSA a la perfecció.

5.3.1 Trusted Virtual Domain (TVD)

El Trusted Virtual Domain és una abstracció lògica que no esta assignat a cap component físic del NED. No obstant, proporciona la manera d'identificar tots els components que estan associats a un usuari. El TVD representa una porció aïllada del NED que es dedica a un únic usuari i que inclou els diferents entorns d'execució, les instàncies de interfícies lògiques associades a un usuari, per implementar tot el data plane i l'encadenament dels serveis que aquest tingui configurat.

El TVD de l'usuari és una partició lògica dins del host, per tant no es pot assignar a un component en concret des de la perspectiva de la implementació. En canvi, la TVD es pot considerar com un conjunt de components que pertanyen a un usuari específic.

5.4 Comunicació entre els elements

La comunicació entre els PSC, PSCM, TVDM i el client es realitzan amb una API REST Gunicorn.
'Green Unicorn' que és un servidor HTTP de Python Web Server Gateway Interface () per a UNIX. El servidor gunicorn és àmpliament compatible amb diversos frameworks web, simplement implementat, baix consum de recursos del servidor, i bastant ràpida.
Entrarem en detall sobre les diferents APIs que s'ha modificat en el capítol de **??**

5.5 Personal Security Controller Manager (PSCM)

El Personal Security Controller Manager és el conjunt de components encarregats de comunicar l'usuari amb el nivell de control (per tant el trànsit el usuaris que serà processat per una PSA no passa a través del PSCM). Internament, està connectat principalment a la xarxa de control i de gestió que també connecta el NED a la SPM i als repositoris.

El PSCM aprofita dos components principals: l'agent de confirmació remot i el mòdul d'autenticació, que s'utilitzen en les primeres etapes de la connexió d'usuari, que són l'establiment de connexió segura entre la terminal de l'usuari i el NED, i l'autenticació d'usuari.

Una vegada aquests passos es realitzen i s'atorga l'accés a l'usuari, el PSCM transfereix el testimoni de la sessió d'usuari al TVDM, que continuarà la creació de la instància TVD de l'usuari. Finalment, el PSCM informa de la resposta inicial de la creació del TVD a l'usuari després de rebre el TVDM.

5.6 Trusted Virtual Domain Manager(TVDM)

L'orquestrador és el component de la presa de decisions que assigna recursos del TVD d'un usuari, crea les connexions d'aquest TVD, els seus connectors i la configuració del seu data plane. No emmagatzema el context de l'usuari, però ha de ser capaç de recuperar fàcilment la topologia interna d'aquest en el NED (per exemple, configuració de la xarxa i les màquines virtuals que tenia instanciades prèviament).

5.7 Network Edge Device (NED)

Aquest component acull tots els components descrits en aquest capítol, i aplicacions de seguretat funcionant concurrentment, proporcionant un punt de control uniforme per a tots els dispositius connectats. Aquest administrador independent, segur i de confiança pot actuar en diversos escenaris diferents, com en passarell·lles residencials, routers d'empresa, punts mòbils o d'accés públic.

L'arquitectura permet a diferents usuaris (per exemple usuaris individuals, administradors de TIC corporatius i xarxa proveïdors) per instal·lar sota demanda i executar Personal Security Applications (PSA) dins del seu propi entorn d'execució de confiança i virtualitzat. El NED allotja i aïlla els entorns d'usuari, així com aïllaments dels fluxos de trànsit del diferents usuaris. Els usuaris són capaços de configurar el seus serveis a través del seu propi () () del NED. Per tant, la seguretat arquitectura de la NED ha de ser acuradament dissenyada per fer front de forma segura les diferents necessitats.

A més del NED, cal esmentar que també que hi ha entitats externes, com ara el gestor de PSAs (PSAM), el repositori de PSAs (PSAR) i el Security Policy Manager (SPM): són els serveis centralitzats dels usuaris descarregar o gestionar els seus serveis contractats i les polítiques de seguretat configurades. Els usuaris podran registrar-se amb i accedir a aquests serveis garantits i ser capaç de triar les polítiques d'alt nivell necessiten per a ser configurades.

Capítol 6

Implementació

Aquesta capítol tractarem totes les modificacions realitzades en el projecte SECURED per dur a terme la implementació de la mobilitat.

6.1 Instanciació

Començarem parlant de la instanciació d'un usuari a l'entorn de SECURED.

6.1.1 Abans d'implementar la migració

En el moment que s'inicia l'etapa de desenvolupament i implementació de la mobilitat del projecte de SECURED, ens torbem davant del següent l'escenari. L'usuari mitjançant, via una interfície web, es pot connectar al NED, es crea un túnel strongswan, per crear una canal de confiança. No obstant, un cop instanciat no existeix la possibilitat de portabilitat i replicació del TVD i la resta de l'entorn de l'usuari que s'acaba d'instanciar. Pot navegar, de forma controlada i segura, només en el NED que fet l'autenticació del seu usuari.

6.1.2 Passos de instanciació del NED

A continuació s'explica de manera detallada, els passos principals d'un usuari per connectarse a Internet mitjançant un NED:

1. **Establir un canal de confiança:** aquest pas consisteix en dues fases: a la primera, l'usuari estableix una comunicació segura amb el NED, en el segon, l'usuari comprova la identitat del NED i els components independents de l'usuari (PSCM, Administrador de directives, TVDM, NED management, etc.).

2. **L'autenticació de l'usuari:** el NED autènticaa l'usuari utilitzant les seves credencials, si l'autenticació té èxit, PSCM rep senyal de sessió posterior.

3. **Sol·licitud d'instàncies TVD:** PSCM demana al TVDM una instància de TVD bàsic per a l'usuari (i l'hi passa el testimoni de la sessió). Inicialment, la TVD inclourà només el PSC i serà llavors s'ampliarà depenent de la gràfica de servei de l'usuari.

4. **Consulta PSC:** el TVDM utilitza l'identificador de sessió a cercar el perfil d'usuari (PSC per la seva secció corresponent).

5. **Creació del TVD i inici PSC:** El TDVM del NED crea el TVD i el PSC de l'usuari s'inicia.

6. **Consulta del perfil de l'usuari:** El TVDM recupera el perfil d'usuari (és a dir, el resum gràfic de servei d'usuari) de la SPM utilitzant el testimoni(token) de sessió.

7. **Traduir polítiques, carregar el perfil d'usuari i configuració de les PSAs:** Si cal, a continuació, just a temps, es porta a terme la configuració de les politiques per l'Administrador de directives. Posteriorment, amb el gràfic d'infraestructura, es fa la configuració del perfil de PSAs i de l'usuari s'envien al PSC de l'usuari.

8. **Obtenir anuncis de servei públic:** Abans de sol·licitar l'expansió TVD, qüestions PSC demanen a la TVDM per al PSA. El TVDM descàrrega PSA de l'usuari utilitzant el token de sessió proporcionada. Mentre que els manifestos de PSA sempre es retornen al PSC, els PSA poden també carregar-se a aquest últim si necessiten ser carregats manualment als TEES.

9. **Es determina la topologia del TVD:** el PSC de l'usuari the in frastructure graph, PSA configurades i manifest per determinar la topologia requerida per a la instanciació del servei de l'usuari.

10. **Sol·licitud d'expansió TVD:** PSC envia una sol·licitud d'expansió del TVD al TVDM per crear noves interficies per a les PSAs. Aquesta petició s'emet a través de la interfície de TVD MGMT.

11. **Ampliar TVD:** TVDM crea els TEEs sol·licitats en TVD de l'usuari, d'acord amb la informació de sol·licitud del TVD d'expansió de l'usuari.

12. **configuracio del TEE :** PSC establirà els tees adequadament i enviar-los a la xarxa configuració (utilitzant la interfície Ctrl / Mgmt). Això és necessari per connectar adequadament els múltiples anuncis de servei públic que s'executen en el mateix ETE.

13. **Configuració a baix nivell de les PSAs :** PSC carregarà PSAs i la seva configuració de baix nivell en els respectius TEE. A continuació, s'iniciarà PSC anuncis de servei públic.

14. **Tipus d'informe:** PSC informa sobre el seu estat a la gestió de la NED i la segona realitza la certificació segona etapa del PSC i anuncis de servei públic. Llavors, la gestió NED genera un informe d'estat global i l'envia al PSCM.

15. **Avaluació d'usuari:** terminal d'usuari rep informe d'estat des del PSCM, verifica informe d'estat està bé, desconnecta la connexió PSCM.

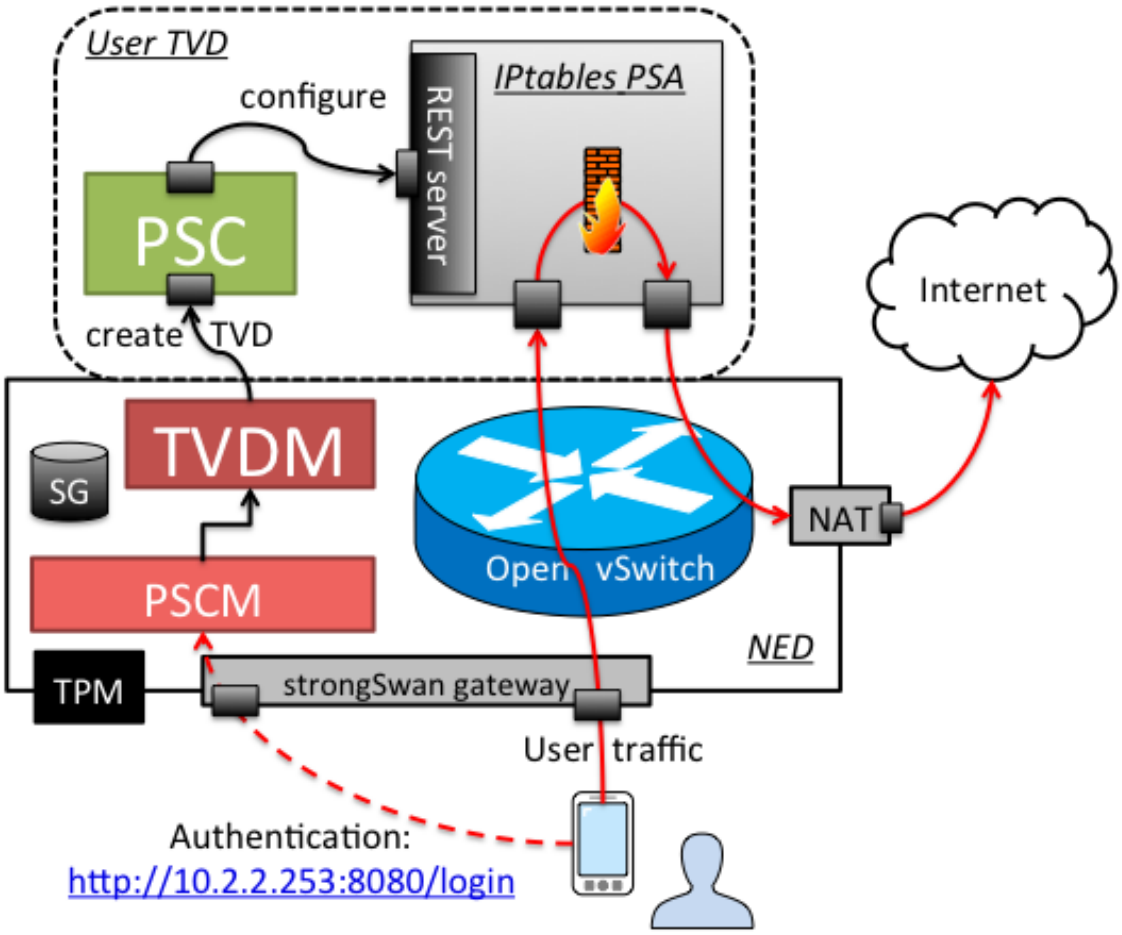


Figura 6.1: Instanciació

6.1.3 Escenari final

Un cop finalitzada la implementació de la mobilitat,de tot el que s'ha explicat en el punt anterior, excepte punt 1 i a més s'afageix un nou punt:

1. Establir un canal de confiança: Ara es realitzara amb una aplicació que funciona en l'equip del client (implementació realitzada per la meva companya Alícia).Juntament amb totes les implementacions noves a StrongSwan, l'aplicació realitza la comunicació amb el NED per iniciar el canal de confiança.

2. Tots els punts, menys el nº1, descrits en la secció anterior.

3. Comprovació de senyals: L'aplicació va mirant totes les senyals dels NED pròxims, i va enviant crides REST al PSC per indicar que si cal o no cal inciar el proces de migració.

Després d'explicar en detall totes les implementacions realitzades a nivell de codi i de funcionalitats internes del NED. Detallarem els següents punts funcionals de la migració.

6.2 Implementacions realitzades per a la mobilitat

6.2.1 Túnel VPN amb StrongSwan

Primer de tot la connexió havia de ser segura, es decideix utilitzar StrongSwan, una solució basada en virtual protocol network. Es va compilar la versió 5.5 amb les següents extensions o plugins:

MOBIKE L'extensió MOBIKE[19] IKEv2 (RFC 4555) permet un túnel ja generat pugui canviar el seu punt d'anclatge de la xarxa. StrongSwan implementa MOBIKE observant les interfícies, adreces i rutes. Si els canvis de configuració, consultes de rutes es realitzen per trobar un camí millor que l'actual i, si cal, el camí es canvia mitjançant una actualització MOBIKE. strongSwan s'ha de canviar al port UDP 4500 a partir per que l'extenció MOBIKE fa us d'aquest port per la comunicació d'aquest port perquè a partir de la sol·licitud de IKE_AUTH, que inclou una notificació MOBIKE_SUPPORTED, encara que no s'ha detectat NAT.

Plugin[?] SQL El plugin de SQL per Charon[?] permet emmagatzemar la configuració de l'enllaç complet en una base de dades relacional. A més, el dimoni llegeix les credencials, com certificats, claus privades o contrasenyes de la base de dades per fer tot tipus d'autenticació. El registre a la base de dades també és possible.Amb aquests dos elements obtenim la manera de preservar el túnel en un canvi de configuració inicial, en el moment de la migració de l'entorn de l'usuari i amb el plugin mysql podem tenir un registre d'usuaris, per suportar múltiples usuaris connectats simultàniament en un sol NED. Generem els usuaris amb les ips que seran assignades. Cada ususari tindrà una ip de rang 10.2.1.x/32.

6.2.2 Funcionalitats de mobilitat afegides a les classes existents

mainIPSEC.py S'ha afegit una ruta per la funcionalitat de migració de la API gunicron del TVD.

Fragment de codi 6.1: Nova ruta de la funcionalitat del TVDM

```
1 app.add_route('/migration', orch)
```

S'ha modificat la ruta de que iniciava la instanciació, amb un paràmetre nou. És el flag d'activació de la migració. Aquest flag controla si s'ha d'intancia tot l'entorn de l'usuari des de zero o ha de regenerar-lo

Fragment de codi 6.2: Afegit nou paràmetre a la ruta de instanciació

```
1 orch = Orchestrator(instantiator, migration)
```

I la instanciació de la migració

Fragment de codi 6.3: Instanciació de la migració

```
1 migration = TVDMigration(instantiator, conf, logger)
```

Orchestrator.py Es l'orquestardor del TVD, es a dir, el ja esmentat TVDM en el capítol 5, es la API de la gunicorn de TVDM. Han sigut afegides noves funcionalitats. Ara quan captura una crida REST POST s'inicia amb la migració de clients amb una crida REST POST que rep del PSC.

Fragment de codi 6.4: Funció on_post()

```
1     def on_post(self, request, response):
2         '''
3         exclusive for migration
4         '''
5         try:
6             self.instantiator.tvdm_receives_migration_request =
                  datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
7             args = request.stream.read()
8             self.TVDMigration.instantiator.logger.info(request.
                  method + " " + request.uri + " " + args)
9             session = json.loads(args, 'utf-8')
10
11            self.eventList[session["token"]] = Event()
12            self.handoverEvents[session["token"]] = Event()
13            migration = Thread(name=session["token"], target=self.
                  TVDMigration.init_migration,
14                               args=(self.eventList[session["token
                                  "]], self.handoverEvents[
                                  session["token"]],), kwargs={"
                                  session": session})
15            migration.start()
16            response.status = falcon.HTTP_200
17        except:
18            self.TVDMigration.instantiator.logger.exception(sys.
                  exc_info()[0])
19            response.status = falcon.HTTP_500
```

També s'ha afegit la funcionalitat que captura crides REST GET per respondre al PSC si ha d'avisar al client si pot canviar de punt d'accés.

Fragment de codi 6.5: Funció on_get()

```
1     def on_get(self, request, response):
2         '''
3         Exclusive for migration, non-bloking query migration
4         '''
5         try:
6             self.TVDMigration.instantiator.logger.info(request.
                  method + " " + request.uri)
7             user = request.get_param("user")
8             action = request.get_param("action")
9
10            obj = {}
11            if action == "migration":
12                if user in self.eventList.keys():
13                    self.TVDMigration.instantiator.logger.info("
                          user: %s, eventList: %s" % (str(user), str
                          (self.eventList[user].isSet())))
14                    obj = {"status": self.eventList[user].isSet()}
15                    if self.eventList[user].isSet() is True:
16                        self.instantiator.
                              tvdm_sends_migration_finished =
                              datetime.utcnow().strftime("%Y-%m-%d %
                              H:%M:%S.%f")
17                        self.handoverEvents[user].set()
18                    else:
```

```
19                          obj = {"status": False}
20              else:
21                  obj["timestamps"] = self.instantiator.timestamps
22              response.body = json.dumps(obj)
23
24              self.instantiator.logger.info(str(response.body))
25              response.status = falcon.HTTP_200
26
27          except Exception as e:
28              self.instantiator.logger.exception(sys.exc_info()[0])
29              self.instantiator.logger.exception(str(e))
30              response.status = falcon.HTTP_501
```

Graphinstantiator.py S'ha afegir el flag de migració, esmentat en les subseccions anterior, per controlar si la instanciació del TVD s'ha de realitzar des de zero o ha de recollir les dades donades per la sessió que rep producte de la migració per regenerar les polítiques d'encaminament Openflow del PSC i PSAs de l'usuari, interficies virtuals'es a dir tot el data plane.

userTVD.py S'ha el flag de migració, aquest flag controla si la configuració del TVD s'ha de generar tot des de zero o ve donada per una migració, per tant només ha de tornar a regenerar les interfícies i les polítiques OpenFlow.

En els annexes B, s'ha marcat amb comentari ###migration code### per indicar les modificacions que s'han realitzat en les classes ja existents.

6.2.3 La nova classe implementada: TVDMigration

A continuació farem una breu explicació d'algunes de les principals funcions de la nova classe implementada que s'encarrega d'efectuar la migració de l'entorn de l'usuari

Funció sendTVD

S'encarrega de preparar en fitxer tot l'entorn de l'usuari que s'ha generat en el NED origen (TVD). Inclou noms de les maquines virtuals (PSAs i PSC), noms de les interfícies virtuals generades, ip assignada al PSC i la configuració del "data plane"de l'uusari. A més a més activa el flag de migació i el guarda també en . I finalment el token de l'usuari.

Tota aquesta informació es introduït en un missatge i s'envia una crida REST a NED destí on rep aquest missatge i en destí te una funció que realitza la re-intanciació de tot aquest entorn.

Fragment de codi 6.6: Funció SendTVD

```
1   def sendTVD(self, session):
2       '''
3       Obtain user's TVD and migrate this
4       '''
5
6       self.TVD['token'] = self.instantiator.userTVDs[session['
            token']].userName
7       self.TVD['IP'] = session['IP']
8       self.TVD['userInterface'] =
```

```
9           self.instantiator.userTVDs[session['token']].
                userInterface
10          self.instantiator.logger.info("userIsessionnterface " +
11           str(self.TVD['userInterface']))
12          self.TVD['vlanID'] = self.instantiator.userTVDs[session['
                token']].vlanID
13          self.TVD['pscAddr'] =
14          self.instantiator.userTVDs[session['token']].pscAddr
15          self.TVD['psc'] = self.instantiator.userTVDs[session['
                token']].psc
16          self.TVD['pscName'] =
17           self.instantiator.userTVDs[session['token']].psc['name']
18          self.TVD['generatedFlows'] =
19           self.instantiator.userTVDs[session['token']].
                generatedFlows
20          self.TVD['PSAs'] = self.instantiator.userTVDs[session['
                token']].psaList
21          self.TVD['psaIPaddresses'] =
22           self.instantiator.userTVDs[session['token']].
                psaIPaddresses
23          self.TVD['migration'] = "True"
24          self.TVD['action'] = "TVD"
25
26          try:
27              TVD = json.dumps(self.TVD)
28              self.logger.info("TVD: " + str(TVD))
29          except:
30              self.logger.info("Error to created TVD json")
31
32          try:
33              cmd = "ip netns exec orchNet curl -X PUT --header \
34              Accept: application/json --header Content-Type:
                    application/json -d\
35               '" + TVD + "' http://192.168.1.1:8080/migration"
36              (results, errors) = self.execute_command(self.ssh, cmd
                    , False)
37              self.instantiator.tvdm_sends_TVD =
38               datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M:%
                    S.%f")
39
40          except:
41              self.logger.info(errors)
```

Funció migration

Inicia el de migració del PSC i les PSAs, aquesta funció llençara diferents theards que estaran controlat per una flag per saber quan ha de llançar el missatge rest amb la configuració de l'usurri, que nomes es podrà fer quan totes les maquines virtuals s'hagin migrat en destí.

La part de codi esta en el annexe B

Funció obtain_disk_base

Obte el disc base de les maquines virtuals

Fragment de codi 6.7: Funció generate_remote_disk

```
1     def obtain_disk_base(self, path):
2         '''
3         obtain path VM disk base
4         '''
5         try:
6             proc = subprocess.Popen("qemu-img info " + path,
7                                     stdout=subprocess.PIPE, shell
                                        =True)
8             (out, err) = proc.communicate()
9             returnedValue = str(out)
10            start = 'file: '
11            end = '\n'
12            disk_base = ((returnedValue.split(start))[1].split(end
                  )[0])
13
14        except subprocess.CalledProcessError as e:
15            self.logger.info("Calledprocerr:" + e)
16
17        return disk_base
```

Funció generate_remote_disk

Comprova primer que existeixi el disc de la maquina virtual en destí sinó el crea i el migra. Aquesta solució es va dur a terme en veure que depèn quins escenaris donava error la migració del disc. Això no comportava tampoc una penalització en temps ja que els disc de les maquines virtuals era molt petit.

Fragment de codi 6.8: Funció generate_remote_disk

```
1     def generate_remote_disk(self, original, disk_type, path, vm):
2         """
3         Generate remote disk
4         """
5         stderr = ""
6
7         try:
8             cmd = "[  -f " + path + "/" + vm + " ] || qemu-img
                  create -b "
9                   + original + " -f " + disk_type + " " + path
10            stdin, stdout, stderr = self.ssh.exec_command(cmd)
11            stdin.close()
12        except Exception as e:
13            self.logger.info("Error to create remote disk " + str(
                  e) + " "
14                             + str(stderr))
```

Funció rename_remote_disk

Si la funció anterior generava el disc dur en remot, aquesta comprova que el nom sigui el de la maquina que es migrarà.

Fragment de codi 6.9: Funció rename_remoter_disk

```
    def rename_remote_disk(self, path, vm):

        stderr = ""

        try:
            cmd = "mv " + path + "/" + vm + " " + path + "/" + vm
                + "backup"
            stdin, stdout, stderr = self.ssh.exec_command(cmd)
            stdin.close()

        except Exception as e:
            self.logger.info("Error to create remote disk "
                             + str(e) + " " + str(stderr))
```

Funció createSSHclient

La funció implementa fa us la llibreria paramiko, per la creació d'un objecte SSHClient. Per a un control més directe, fa us d'un socket (o un objecte socket-similars) per al transport, i utilitza start_server o start_client a negociar amb el host remot com un servidor o client. D'aquesta manera es comuniquen els NED entre ells per iniciar, mantenir i finalitzar la migració de les maquines virtuals.

Fragment de codi 6.10: Funció createSSHclient

```
    def createSSHClient(self, server, port, user, password):

        client = paramiko.SSHClient()
        client.load_system_host_keys()
        client.set_missing_host_key_policy(paramiko.AutoAddPolicy
            ())
        client.connect(server, port, user, password)
        return client
```

Super classe TVDMigration

Aquesta classe que executa la migració de les maquines virtuals (PSASs i PSC). Amb l'us de la llibreria libvirt, s'inicia la connexió amb el NED remot, mitjançant l'us del [**?**] creat amb la funció createSSHclient, explicada anteriorment, un cop esta establerta, inicia la migració en calent de la maquina virtual. Un cop finalitzada, s'elimina la maquina en origen que s'ha quedat en , per evitar un consum excessiu de memòria.

La part de codi de la super classe esta en el annexe B

6.2.4 Noves funcionalitats incloses en les API REST gunicorns

Tal com es va explicar en el capítol de l'arquitectura, en el NED esta corrent múltiples API REST gunicorns. Per implementar la migració es va implementar un seguit de noves funcionalitats. La comunicació entre el client i el servidor també es feia mitjançant captura de missatges REST JSON.

Les extensions de mobilitat a la NED cobreixen bàsicament tres parts diferents, i) les interfícies de migració de màquines virtuals en el PSCM, ii) de l'API de migració d'estat de la xarxa en el TVDM i iii) de l'API d'informes del senyal al PSC.

El flux de treball principal de la migració és el següent:

1. El client informa de les senyals WiFi del punts d'accés. PSC avalua l'estat del senyal, que es correlaciona amb el passat. En el cas que no cal migrar, els rendiments del sistema.

2. Si la migració és necessari, el sistema invoca el procés de migració les maquines virtuals dins el PSCM.

3. Al mateix temps que sol·liciten la migració de l'estat de la xarxa TVDM.

4. S'informa al node mòbil per disparar el canvi de túnel.

canvis en l'API del TVDM La migració TVD es basa en la replicació de la configuració de xarxa en els complements sistema de destinació realitzades en el TVDM proporciona la lògica necessària per a la migració de la TVDM a l'altra NED. Amb aquesta finalitat, afegim una nova crida a l'API: la migració, que assiteix en la migració TVDM, i un altre ha de captura la migració, *instantiateTVD* , que proporciona capacitats de re-instanciació d'un TVD. L'explicació detallda del codi afegit esta en la subsecció 6.2.2 .

Canvis en l'API per al PSC Finalment, per permetre que el client per proporcionar informació sobre la intensitat del senyal del node mòbil hem proporcionat un nou mètode que permet que aquests informes, i també proporciona una resposta adequada al client amb dues opcions:

- FORCE_HANDOVER: permet que el client realitzi la migració, si cal, tan aviat com és possible.

- NO_HANDOVER: Opció bloquejant, de manera que el client ha de romandre associat en el mateix punt d'accés.

6.2.5 Parall·llelitazació i control fluxe d'execució de la mobilitat

Com bé es va explicar en el capítol de planificació. Un cop provat que la integració de la migració funcionava,es va dur a terme l'optimització d'aquesta. La solució triada, amb la classe Events de la llibreria Threaring. Quan es fa la crida d'un Event(), aquesta funció que retorna un nou objecte Event. Un Event es gestiona mitjançant una flag, per defecte es fals. I es posa a veritable amb el mètode Event.set(). El mètode Event.wait(), un es mante bloquejat fins que l'indicador de la flag sigui veritable.

Per optimitzar la migració, fem servir els events per controlar el fluxe d'execució de la mobilitat. Com ja s'ha vist anteriorment ja es fa servir threads per iniciar la migració. No obstant el thread es únic per cada usuari i es iteratiu. Ara s'incorpora un event per saber quan acaba la migració de les PSAs i passar el testimoni al client per fer el canvi de NED. Per tant un cop creat l'event, el guardem una llista de hash d'events, la clau sera el testimoni de l'usuari. Un cop guardat es genera una instància de migració,on passem els paràmetres de sessió(on hi ha el testimoni de l'usuari) i l'event.

Dins de la instància de migració de cada usuari, s'inicia la migració de totes les PSAs de forma parall·llela, cada una generara un event, Que es guardara en llista. Després de llançar-se tots els threads de migració de les PSAs.
Es preparà un thread per migrar el PSC, no començarà a migrar fins que totes les PSAs no hagin migrat. També crearà un thread d'enviament del TVD del l'usuari. Aquest thread

quedara bloquejat a l'espera de que s'hagi iniciat la migració del PSC, llavors s'enviarà el TVD.

Mentre es va executant EL thread principal de migració, el PSC va llançant crides REST on_get(), aquest procés es repeteix, fins que l'event inicial no es posa a veritable.Cada una d'aquestes PSAs també te un event, per tant, quan totes els events de les PSAs, son veritables, es desbloqueja el event principal. En aquest punt es desbloqueja el PSC per enviar el missatge de confirmació de canvi de punt d'acces al client.

Finalitzaran tots els threads i s'allibera l'espai de memòria amb un join() general. Per assegurar que cap thread queda obert.

6.3 Escenari final: Migració

1. **PSC captura missatge de migració del client:** El PSC captura el crida REST d'inici de migració, passa el testimoni al TVDM amb una variable sessió que conte entre altres coses la clau de l'usuari i la ip del NED on ha de migrar.

2. **TVDM captura el missatge del PSC:** Oquestrador del TVDM rep un missatge REST on_post() amb la variable sessió, genera el event de migració de l'usuari i llança el thread de migració.

3. **Migració PSAs:** Primer començar a migrar-se totes les PSAs de l'usuari

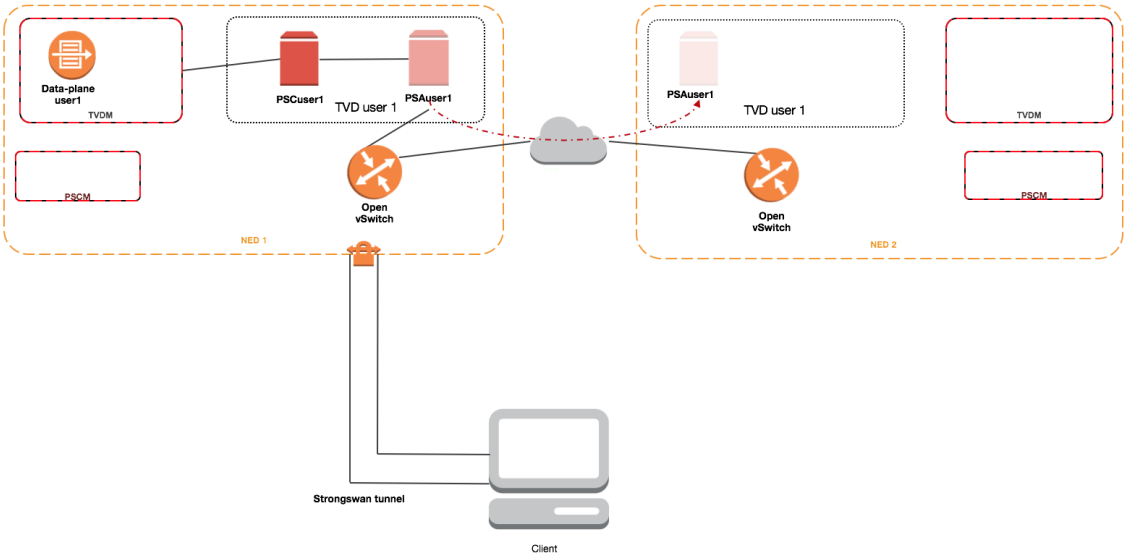


Figura 6.2: Migració de les PSAs

4. **Migració PSC:** Quan la migració de totes les PSAs esta al 80%, s'inicia la migració del PSC

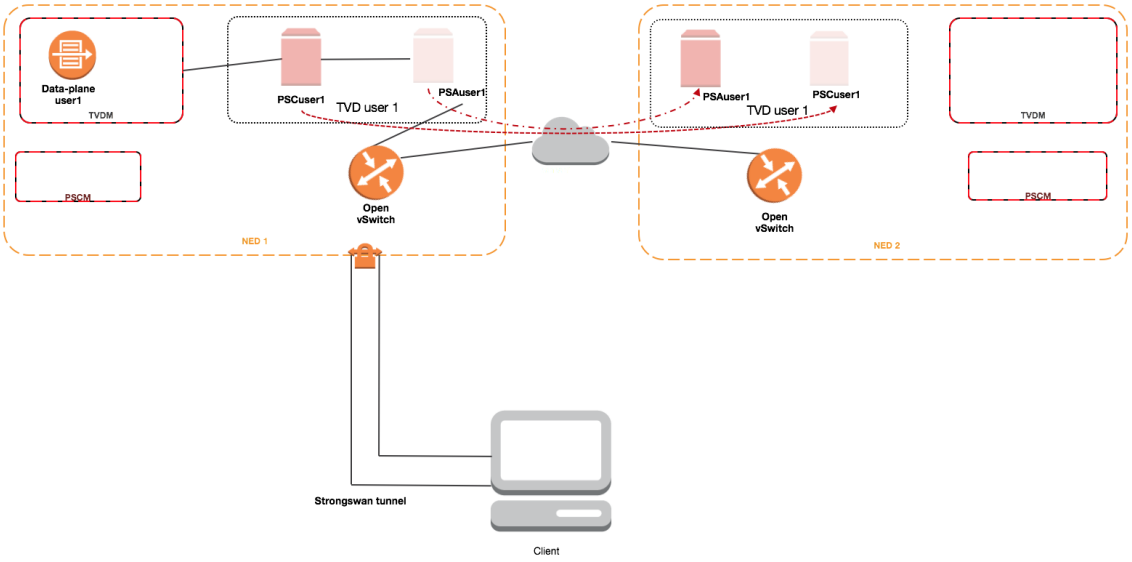


Figura 6.3: Migració del PSC

5. **TVDM respon al PSC:** El PSC un cop s'ha iniciat la migració, envia crides REST on_get() per rebre una resposta positiva del TVDM. Quan el TVDM respon, el PSC recull el testimoni.

6. **PSC respon al client:** EL PSC respon al client que ja pot fer el canvi

7. **Enviament del TVD:** Instants després d'iniciar la migració del PSC, s'envia amb un missatge REST, un json amb tots el TVD del l'usuari per ser re-instanciat en destí.

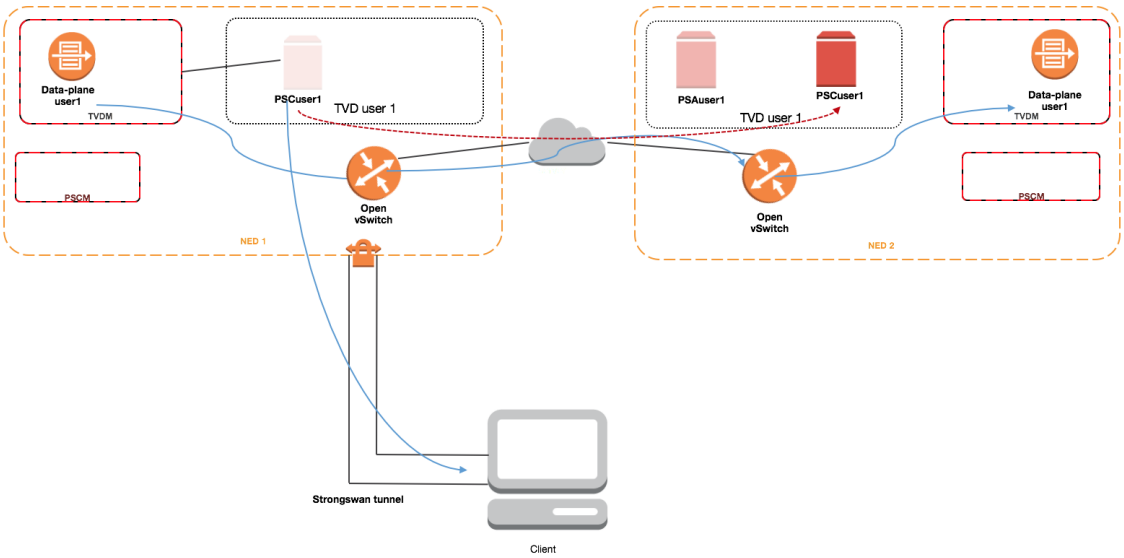


Figura 6.4: Enviament del TVD

8. **Re-instanciació de tot l'entorn:** Es re-instància tot el data-plane de l'usuari, es regeneren totes les interfícies de l'usuari i la seva configuració.

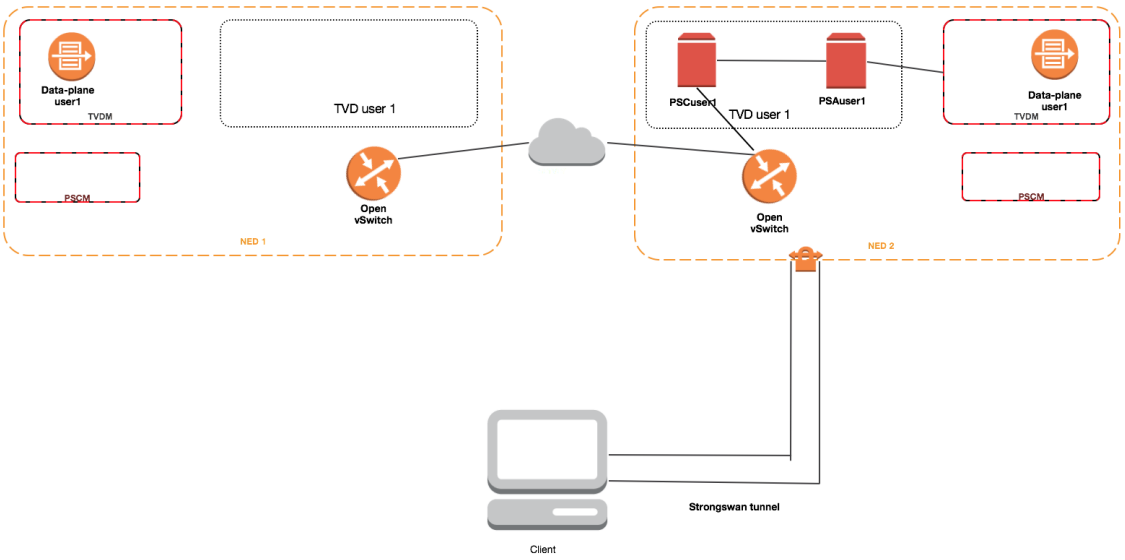


Figura 6.5: Re-instanciació del TVD

9. **El client canvia de AP i el punt final del túnel:** El client baixa el túnel StrongSwan del NEd origen, es connecta al nou punt d'accés i aixeca el túnel en el NED de destí.

Capítol 7

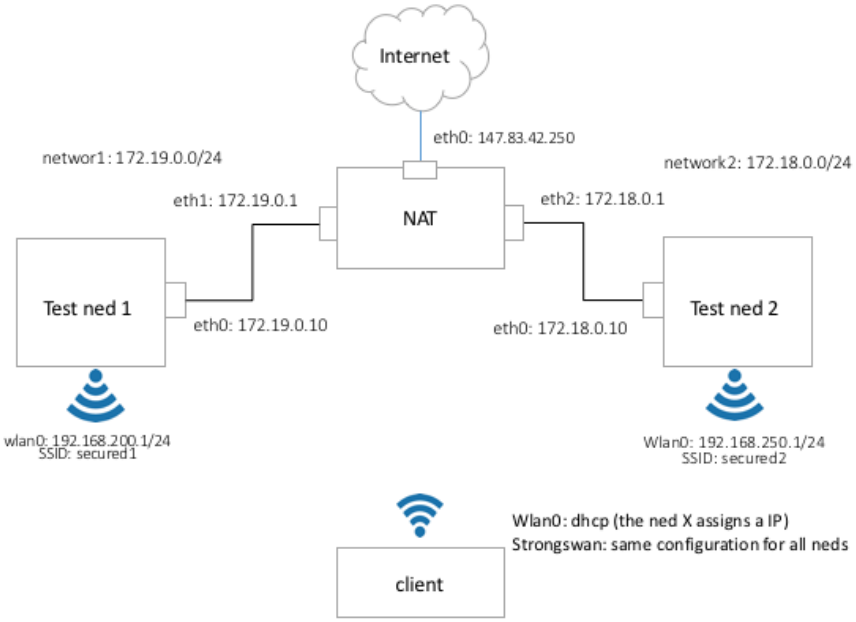
Proves de rendiment

En aquest capítol parlarem de l'entorn que s'ha configurat per realitzar les proves de rendiment de la implementació de mobilitat realitzat al projecte SECURED.

7.1 Banc de proves

L'entorn configurat consta de dos equips en format [17] i un portàtil, els dos nucs estan connectats entre elles per una xarxa local, amb una única sortida a Internet, tal com vèiem a la figura 7.1.

Figura 7.1: Disseny conceptual del banc de proves



Com es un disseny conceptual. Ja que l'element NAT no es mes que un glsgunicorn corrent en en el test ned 1. Aquesta maquina físicament es l'únic que te sortida a Internet. Per tant cada cop que s'efectua el canvi d NED, rep una "sol·licitud amb la informació necessària per saber per on ha de re-direccionar transit cap a un ned o un altre. Es va prendre aquesta solució per estalviar recursos i afegir una tercera maquina física , amb la despesa econòmica i energètica comportava, només per realitzar aquesta funció. Finalment

tenim la taula següent amb la informació bàsica de maquinari i programari instal·lats en les maquines que es fan servir per les proves.

Equips			
Nom	test ned 1	test ned 2	client
Sitema/fabricant	NUC5i5RYBAF	NUC5i5RYBAF	ASUS
CPU	Intel Core i5-5250U	Intel Core i5-5250U	Intel Core i5-2430M
Memòria	8GB	8GB	4GB
Distribució	Debian GNU/Linux 8		
	3.16.0-4-amd64	3.16.0-4-amd64	4.7.0-1-686-pae

Taula 7.1: Característiques dels equips utilitzats

7.2 Funcionalitats afegides per la realització del test

S'ha afegit la llibreria datetime, fent us de la funció:

```
1 |     datetime.utcnow().strftime(\"\%Y-\%m-\%d \%H:\%M:\%S.\%f\")
```

On ens retorna aquesta data amb aquest format: 2016-11-11 09:37:50.391198

Amb aquesta funció hem recopilat el temps exacte de les accions dels elements que intervenen en la migració, sense notar cap penalització de rendiment.
Per tant s'ha afegit timestamps en tot el fluxe d'execució, des de que el client reporta que ha de migrar fins que el túnel StrongSwan s'ha tornar a crear en destí:

1. CLIENT SENT REPORT: Instant de temps en que el client envia un missatge al PSC que ha d'iniciar el proces de migració
2. PSC RECEIVED REPORT: Instant de temps en el que PSC rep el missatge del client.
3. PSC SENT MIGRATION REQUEST: Instant després de la captura el missatge del client envia la sol·licitud d'inici de migració al TVDM.
4. TVDM RECEIVED MIGRATION REQUEST: Instant de temps en el TVDM rep la solicitud de migració
5. TDVM PSA INI/FIN: LEs següents marques es realitzen en paral·lel, per tantes PSAs tingui instanciades l'usuari
   (a) TVDM PSA INI: Instant de temps en el que inicia de la migració de n PSA.
   (b) TVDM PSA FIN: Instant de temps en el que finalitza la migració de n PSA.
6. TVDM PSC INI: Instant de temps que el PSC inici la seva migració cap al NED de destí
7. TVDM SENDS TVD: Instant de temps en el que el TVDM envia el TVD de l'usuari
8. TVDM PSC FIN: Instant de temps que el PSC ha acabat la seva migració cap al NED de destí

9. TVDM FINISHED MIGRATION: Instant de temps que ha acabat tot el proces de migració per part del NED origen
10. TVDM SENT MIGRATION FINISHED: Instant de temps en El TVDM envia un missatge de finalització
11. TVDM2 RECEIVED TVD: Instant de temps en que el TVDM del NED destí rep el TVD que ha enviat el TVD del NED origen.
12. TVDM2 FINISHED INSTANTIATION PSAs: Instant de temps en el que acaba de re-instanciar totes les PSAs en el NED de destí
13. TVDM2 FINISHED INSTANTIATION PSC: Instant de temps en el que acaba el PSC de re-instanciar-se en el NED de destí.
14. PSC RECEIVED MIGRATION FINISHED: Instant de temps en el que el PSC re-instanciat rep el missatge de finalització de la migració, enviat previament pel TVDM del NED origen.
15. PSC SENT HANDOVER ORDER: Instant en el que just després de rebre el missatge de finalització de la migració, envia l'ordre al client per faci el "handover".
16. CLIENT RECEIVED HANDOVER ORDER: Instant en el que envia l'ordre al client per faci el "handover".
17. CLIENT CLOSED TUNNEL: Instant en el que el client treu el túnel amb el NED origen.
18. CLIENT CHANGED APS: Instant en el que el client esta fent el canvi de punt d'accés (es desconnecta del NED origen i es connecta al NED destí)
19. CLIENT CREATED TUNNEL: Instant en el que el client aixeca el túnel amb el NED destí. I aquí es finalitza tot el proces de migració

7.3 Execucions

La prova que s'ha realitzat s'ha fet amb un usuari amb dos PSAs instanciades, una amb el servei de VPN corporativa i una segona amb el servei de tallafocs. S'ha iniciat la recoll·lecció de dades quan ja portava realitzades un total de 5 migracions entre els dos NEDs.

Cada marca de temps, fa la crida a la funció ja mostrada anteriorment i llavors el client recull aquests marques i genera un fitxer resultant amb totes les dades. S'ha realitzat unes múltiples execucions, en el llarg d'un mes i s'han triat grups de 5 all·lleatoriament, per aproximar al màxim la fidelitat de les dades i no tinguem outlier que poguessin diferir molt. Els fitxers resultants es poden veure en els Annexes.

En el següent capítol mostrarem els resultats i els analitzarem.

Capítol 8

Resultats

8.1 Gràfica de resultats

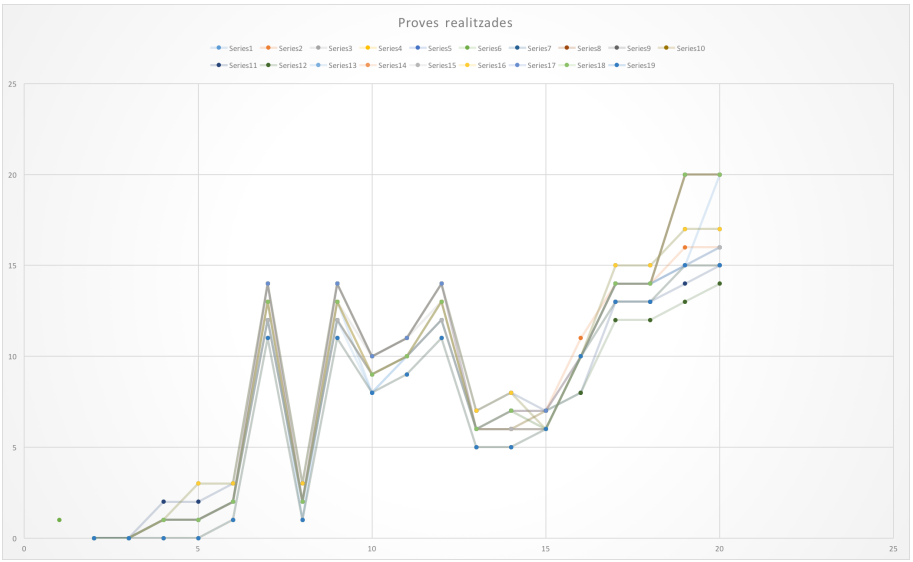


Figura 8.1: Resulats de les proves

La gràfica son totes proves realitzades, en el eix de les X tenim el temps i el eix Y sónels timestamps que s'han coll·llocat en el codi, estan descrites en el capítol 7

8.2 Interpretació dels resultats

Com podem observar en els resultats, podem assegurar que el temps màxim de migració es d'uns 17 segons. Si s'escala la quantitats de PSAs Afectaria mínimament, ja que es realitza parall·llelament. Totes les proves s'han realitzat mentre el client estava visualitzant un vídeo de Youtube.
En ningún moment s'ha tallat el vídeo, la mobilitat ha esta en funcionament tota l'estona i el portàtil s'anava desplaçant deliberadament per tota la zona on estava el banc de proves per forçar la migració.

Capítol 9

Sostenibilitat

En aquest capítol es veurà l'informe de sostenibilitat. El que aquest pretén mostrar és una anàlisi de tots els efectes i despeses que generarà el projecte SECURED abans, durant i posteriorment del seu desenvolupament. Com afectaran aquest en àmbits tant socials, econòmics com ambientals per a poder-los valorar i considerar dins del projecte.

9.1 Anàlisi de sostenibilitat

Sabem que la sostenibilitat és un dels principals reptes d'aquest segle. Ens estem conscienciant dels efectes negatius que tenen les nostres accions sobre l'entorn, sabem de l'existència dels límits que té el planeta i de les injustícies socials que es cometen cada dia, igual que de la importància de treballar de manera sostenible. Tot i així el concepte de sostenibilitat segueix sent una matèria pendent, difícil de concretar i acotar.

És per aquest motiu i per a poder realitzar un estudi de sostenibilitat correcte del projecte SECURED s'ha decidit utilitzar una matriu de puntuacions que s'obtenen a partir d'un llistat de preguntes que corresponen a cada part.

L'anàlisi que s'efectua amb aquesta matriu **??** consta de tres parts: la part de posada en marxa del projecte, la part de vida útil i resultats que té i per últim els riscos que poden aparèixer. Alhora, cada una d'elles s'analitzen des de les tres dimensions de la sostenibilitat: l'econòmica, la social i l'ambiental.

Per a la part posada en marxa del projecte, en la part que ens pertoca de l'equip UPC, mirant les tres dimensions, el que es busca és saber si s'ha considerat l'impacte de realització del projecte, en l'apartat de mobilitat. Tant per aquelles beneficiaris, per aquells que el gestionen, per al planeta i també per a la nostra limitació pressupostaria. Saber si s'ha estimat correctament i s'ha quantificat el cost que pot implicar i si s'han pres les mesures necessàries per a reduir aquest impacte.

En la part de vida útil del projecte es pretén veure com es resol en l'actualitat el problema a resoldre (com bé s'ha descrit prèviament en el capítol Abast) i de quina manera millorarà la situació actual la implantació de la nostra solució. També es vol saber quin serà l'impacte que produirà durant la seva vida útil i quin tipus d'impacte pot produir el projecte a l'hora del seu desmantellament.

Per últim, en la part de riscs el que es vol és aconseguir enumerar aquells escenaris que

són factors de perill per augmentar els impactes negatius que pot produir la consecució del projecte i poder tenir un pla d'actuació o saber de quina manera es podrien mitigar aquests efectes tant en la dimensió ambiental amb la petjada ecològica, en la dimensió econòmica amb la viabilitat del projecte o en la dimensió social amb el perjudici d'algun sector particular de la població o la creació de dependències.

I aquí tenim la matriu de sostenibilitat d'un projecte, de la que tant parlem, amb el seu rang de puntuacions possibles inclòs:

	Ambiental	Econòmic	Social
Projecte en producció	Anàlisi de recursos	Viabilitat econòmica	Impacte personal
Valoració:	[0:10]	[0:10]	[0:10]
Vida útil i resultats	Petjada ecològica	Cost final	Impacte social
Valoració:	[0:20]	[0:20]	[0:20]
Riscs	Perjudicis ambientals	Riscs econòmics	Perjudicis socials
Valoració:	[-20:0]	[-20:0]	[-20:0]
Valoració:	[-60:90]		

Taula 9.1: Matriu de sostenibilitat i puntuacions possibles

Les puntuacions s'obtenen a partir de les reflexions i respostes generades per la bateria de preguntes que es correspon a cada casella. Aquesta matriu s'ha plantejat per a que es pugui fer servir en qualsevol projecte d'enginyeria. Les seves respostes donen lloc a l'informe de sostenibilitat del projecte.

Qualsevol persona responsable d'un projecte ha de conèixer i analitzar el seu projecte amb profunditat, saber on s'emmarca i els efectes derivats de la seva implantació i desenvolupament. Considerem que aquest anàlisi té un pes important en el projecte i que és important recalcar la sotenibilitat del projecte dins el marc de la cooperació per al desenvolupament.

A partir d'aquí es realitzarà una discussió amb la intenció de donar les justificacions adients i poder realitzar una puntuació correcta per a cada casella de la matriu. Es farà per a cada part definida, la posada en producció del projecte, la vida útil i els riscs possibles i s'acabarà amb una avaluació final i les conclusions d'aquest anàlisi.

Per a titllar un projecte de sostenible s'ha d'aplicar una visió global que inclogui les tres dimensions de sostenibilitat (ambiental, social i econòmica) i on s'hi vegin les relacions existents entre les tres components. Per això s'analitzaran les dimensions i les seves interrelacions de forma comú en les tres parts.

9.2 Projecte en producció

Les preguntes a les que es vol respondre per a realitzar el plantejament de la sostenibilitat en la posada en producció del projecte són les que us presentem a continuació:

Dimensió ambiental:

- S'ha estimat l'impacte ambiental que tindrà la realització del projecte? Es pot minimitzar re- utilitzant recursos?
- S'ha quantificat l'impacte ambiental? Quines mesures s'han pres per a reduir-lo?
- Quina és la procedència de les matèries primeres usades? El seu origen o fabricació és ètic?
- S'ha tingut en compte el desmantellament una vegada acabi la vida útil del projecte?

Dimensió econòmica:

- S'ha estimat el cost de realització del projecte?
- S'ha quantificat aquest cost? Quines decisions s'han pres per a reduir aquest cost?
- S'ha ajustat al cost previst? La inversió inicial permetrà que sigui competitiu.

Dimensió social:

- Quines aportacions té realitzar el projecte a nivell personal?
- Ha implicat reflexions significatives a nivell personal, professional o ètic en les persones que han intervingut?
- Quina és la situació social i política del país on s'implantarà?
- L'activitat realitzada afavorirà o empitjorarà aquesta situació?

Tot projecte d'enginyeria neix de la necessitat de resoldre un problema o de materialitzar una solució. L'entorn de SECURED que es vol generar, ha de ser robust, econòmicament viable, ha de tenir en compte el fàcil accés i l'us dels usuaris i ha de tenir un cost ambiental sostenible. Qualsevol projecte que hagi de ser portat a terme doncs, ha de ser subjecte a un estudi de viabilitat.

Pel que fa al projecte SECURED, després d'haver fet una avaluació exhaustiva dels recursos materials i humans en el Capítol del pressupost, disposem d'una xifra estimada. Aquesta, manca de sentit si no relacionam aquests recursos amb l'impacte real que tenen en l'entorn. Però primer de tot, hem de tenir en consideració que aquest és un projecte de cooperació entre diferents universitats i empreses de telecomunicació i que ha estat finançat per la Unió Europea, no obstant cada equip rep un pressupost varitat, l'equip UPC es dels mes modestos. Un dels objectius primordials ha estat el que les despeses econòmiques fossin les mínimes possibles.

Donat aquesta situació, s'ha de tenir en compte que el cost estimat que s'ha presentat no és un cost econòmic real ja que gran part d'aquesta despesa s'ha realitzat en forma d'inversió, bé sigui en hores de treball i dedicació, en la disponibilitat per viatjar o en forma de donació de material informàtic en desús per part de l'equip que han deixat el seu granet de sorra aportant un portàtil o components variats.

És molt important remarcat que dins l'objectiu de dotació **tecnològica** la infraestructura aportada per les proves es nova, exceptuant el portàtil que s'ha utilitzat com a client.

Però un cop finalitzat aixo pot ser replicat en infraestructures ja en funcionament. Això redueix l'impacte negatiu considerablement en les tres dimensions. Socialment, els beneficiaris no els dificultara el canvi de manera de conectar-se a la xarxa. Econòmicament es produeix una petita inversió dels recursos humans a l'hora de la cerca i posada a punt, però n'elimina gairebé la totalitat d'inversió econòmica. S'ha pagat la majoria del material per utilitzar-lo en aquest projecte, no obstant serà utilitzat a molt d'altres, una vegada que aquest conclogui.I finalment el consum elèctric durant el seu ús.

Pel que fa al tema **ambiental**, hi podríem dedicar un capítol sencer a l'impacte positiu de l'entorn, però hem volgut resumir-ho esmentant el més important. El fet de que l'entorn es vitualitzat en la seva totalitat, amb l'us d'ordinadors actuals, sense haver de comprar més aparells. Això en redueix moltíssim l'impacte de triar una solució diferent: s'eliminen els grans costs i recursos necessaris per la construcció de nova tecnologia.

En definitiva, quan decidim que la solució sigui un entorn virtualitzat. fem una contribució a disminuir la nostra petjada ecològica i no per això disminuïm la qualitat ni solvència de la solució trobada.

Socialment tindrà un gran impacte positiu perquè, encara que l'utilitzi algun grup reduït, la seguretat que aportara l'entorn de SECURED ajudara a protegir les dades sensibles de possibles o que es corrompin per la infecció d'un virus o la possible subtracció d'aquesta, estarem ajudant a totes aquestes persones a poder-se connectar de forma segura.

Totes aquestes solucions i més reflexions s'han generat durant el procés de desenvolupament del projecte. Aquest és l'origen de l'impacte personal que he rebut a partir que hem vaig posar a donar suport en l'arquitectura i la mobilitat del projecte SECURED. Aquest projecte és de recerca i cooperació, es situa un entorn molt diferent al d'una empresa amb finalitats lucratives, per tant la corba d'aprenentatge es elevada. Fins que no s'acaba el desenvolupament no veus com els resultats podrien afectar positivament a la societat. Sobretot el que s'ha apres més, es la necessitat de saber **adaptar-te** a diferents circumstancies de fer tot i que no siguin del teu grat i a qualsevol imprevist que hagi esdevingut. L'impacte que m'emporto són tots els coneixements que he obtingut i els aprenentatges dins i fora de la carrera, no podrien tenir millor escenari.

9.3 Vida útil i resultats

Les preguntes a les que s'intenta respondre per a la vida útil del projecte són les següents:

Dimensió ambiental:

- Com es resol actualment el problema? En que millorarà ambientalment la nostra solució de les existents?
- Quins recursos s'usaran durant la vida útil del projecte? Quin serà l'impacte d'aquests?
- El projecte permetrà reduir l'ús de recursos? Globalment, millorarà o empitjorarà la petjada ecològica?

Dimensió econòmica:

- Com es resol actualment el problema? En que millorarà econòmicament la teva solució?

- Quin cost estimat tendrà el projecte durant la vida útil? Es podria reduir aquest cost per fer-lo més viable?
- S'han tingut en compte el cost de les possibles actualitzacions durant la vida útil del projecte?

Dimensió social:

- Com es resol actualment el problema? En que millorarà socialment la teva solució?
- Existeix una necessitat real del projecte?
- Qui es beneficiarà de l'us del projecte? Algun col·lectiu en pot sortir perjudicat?
- De quina manera soluciona el projecte el problema plantejat inicialment?

Com s'ha dit en l'apartat anterior, aquest projecte neix d'unes inquietuds d'aportar, amb un el nostre entorn, una manera segura de connectar-se a la xarxa. Per a complir això cal que un equip suporti el nou entorn, per això es pot re-utilitzar el que es tingui o fer ús d'un que ja estigui en el mercat i re-utilitzar-lo .

Les solucions proposades amb el projecte SECURED contemplen un estalvi futur en temps i diners. Perquè un cop estigui configurada la solució, els beneficiaris només, mitjançant l'aplicació es connectaran a la xarxa mitjançant la nostra solució i no s'hauran de preocupar de res més. No caldrà fer més inversions en el temps que pugui estar en funcionament.

L'impacte mediambiental també es veurà reduït al mínim, perquè tot i tenir una petita despesa energètica per l'us d'un sol equip físic, es redueix considerablement l'emissió de gasos d'afecte d'hivernacle. El fet que es pugui re-utilitzar l'equip, si aquest ho permet, allarga el cicle de vida del l'ordinador. Si no es el cas la inversió es mínima i poden ser equips ja fabricats i es poden re-utilitzar. Doblant el temps d'ús estem estalviant l'energia i recursos necessaris per la fabricació i comercialització de nous equips. Això significa que també s'evita la creació de residus electrònics, altament tòxics.

L'impacte social és pot mesurar amb l'aportació de seguretat i tranquill·llitat dels beneficiaris en la navegació segura, fent us de la nostra solució.

9.4 Riscs

Per a l'ultim bloc de riscs els presentem la bateria de preguntes a les que es vol respondre a continuació:

Dimensió ambiental:

- Es podrien produir escenaris que fessin augmentar la petjada ecològica del projecte? Es pot prevenir o mitigar l'impacte d'aquests possibles escenaris?

Dimensió econòmica:

- Es poden produir escenaris que perjudiquessin la viabilitat del projecte? Es pot prevenir l'impacte d'aquests possibles escenaris?

Dimensió social:

- Es poden produir escenaris que fessin que el projecte perjudiqués a alguna part de la població? Es podria prevenir o mitigar aquest impacte de possibles escenaris?
- Podria crear el projecte algun tipus de dependència que deixés als usuaris en posició de debilitat?

Tot projecte té uns riscs inherents que poden aparèixer, perquè en són molts els factors que influeixen i moltes les situacions a les que ens hem d'adaptar durant la seva possible introducció en una entorn real.

En la dimensió econòmica un dels riscs que provocaria un impacte més important seria que es tingues que comprar un nou equip per incorporar el NED, ja que l'equip que es disposa no es compatible. Implicaria una inversió en l'adaptació del nou També en podrien existir d'altres riscs que perjudiquessin la viabilitat del projecte, però el fet que aquest projecte de cooperació estigui pensat amb la voluntat de re-utilització de recursos li dóna molta sortida davant la necessitat d'adaptar-se a noves situacions.

En la dimensió més social, els beneficiaris tant empreses com a particulars,hauria de ser una operadora de telefonia que s'encarregues a oferir aquest servei. Una dependència, que queda en mans de les les operadores que treballen en el projecte, han de valorar si finalment els es rentable introduir aquest sistema en algun paquet de solucions que poden oferir als seus clients.

I ambientalment, un risc que hi ha és el que les operadores es desentengui totalment dels equips, si es el cas, on esta funcionant la solució proposada. Una vegada deixin de donar suport, els clients llençarien els equips a les escombraries. On puguin contaminar enlloc de seguir el procés de retorn i reciclatge. Per la resta d'objectius, no hi ha un perill evident ja que si no es fa us d'un equip físic, l'entorn es pot virtualitzar en equips ja en us.

9.5 Avaluació de la sostenibilitat

Una vegada presentada la matriu, vistes les preguntes i realitzada la discussió, només ens falta avaluar-la i posar una nota que sigui el resum de les raons per a que aquest projecte pugui ser considerat com un projecte sostenible. Tot seguit ensenyam la matriu completada:

	Ambiental	Econòmic	Social
Projecte en producció	Anàlisi de recursos	Viabilitat econòmica	Impacte personal
Valoració:	9 [0:10]	8 [0:10]	8 [0:10]
Vida útil i resultats	Petjada ecològica	Cost final	Impacte social
Valoració:	15 [0:20]	13 [0:20]	12 [0:20]
Riscs	Perjudicis ambientals	Riscs econòmics	Perjudicis socials
Valoració:	-6 [-20:0]	-6 [-20:0]	-15 [-20:0]
Valoració:	40 [-60:90]		

Taula 9.2: Matriu de sostenibilitat i puntuacions possibles

Les puntuacions que s'han donat en cada casella són fruit de les valoracions argumentades en l'apartats anterior, posant sobre la balança els efectes i costs, tant positius com negatius a considerar de les influències que afecten el projecte en aquella dimensió. Per a

començar, en l'apartat de projecte en producció, dintre d'un rang de puntuació que va del 0 al 10, totes les dimensions han obtingut puntuacions ben elevades. En general això és degut a que, com a projecte de cooperació que és, l'àmbit social aquí arriba a ser més un objectiu que sols una dimensió a avaluar, doncs els costs i la petjada ecològica no tindran sentit si no s'aconsegueix, aquell objectiu social de cooperació.

Per aquest motiu concloem amb la puntuació següent:

La dimensió ambiental obté un 9, l'econòmica un 8 i la social un 8.

El següent bloc de vida útil del projecte pot tenir unes valoracions que van des de 0 al 20 punts. Veiem que, en el nostre cas, totes les dimensions tenen una valoració positiva, doncs creiem que la resolució del nostre projecte ha tingut un impacte real dins el plantejament inicial i a com es solucionava abans. Les solucions proposades amb el nostre projecte contemplen un estalvi futur en temps i diners i no impliquen una despesa de manteniment excessiva. L'impacte mediambiental es veu reduït a la mínim expressió i es té en compte des de l'inici, el possible desmantellament del projecte i els seus materials. I pel que fa a la part social, aportant una tanquill·llitat en mantenir segur l'entorn de navegació del beneficiari, evita el fet de ocasionar pèrdua de dades o robatori d'aquests. Es pot dir que tindrà impacte positiu si s'aconsegueix. Per aquest motiu concloem amb la puntuació següent:

La dimensió ambiental obté un 15 sobre 20, l'econòmica un 13 i la social un 14.

En l'ultim bloc de riscs només es poden tenir valoracions negatives, ja que qualsevol risc és un perill que pot afectar negativament al projecte. Existeixen uns riscs importants i tots ells s'han de tenir en compte i veure quins poden suposar unes pèrdues massa grans, en el cas d'aparèixer.

Hi ha el risc de que apareguin imprevistos que suposin costs econòmics grans, o el risc socials, que es el més important,que no es pugui introdir correctament o No sigui una solució que agradi als possibles beneficiaris. Per ultim el risc ambiental esta present si tots les maquines fisiques que s'utilitzen acaben sent substituides per una nova solució, seran demantellades i generaran tot un seguit 'elements contaminants pel medi ambient. Per aquest motiu concloem amb la puntuació següent:

La dimensió ambiental obté un -6 sobre -20, l'econòmica un -6 i la social un -15.

Finalment resumim dient que per a aquest projecte la puntuació ha estat de 40 sobre 90. Considerem aquesta puntuació una xifra bona, tot i que al risc social es elevat. Conclourem doncs que l'anàlisi de sostenibilitat defineix el projecte SECURED es sostenible.

9.6 Conclusions

Un projecte de recerca cooperatiu saber, intenta millorar certs aspectes socials i economics que afecten negativament a la societat. PEr aquesta premisa es mira si es factible la realització d'una solució per aportar un impacte tant positiu com negatiu en un entorn real.

S'ha demostrat, en el cas de Mallorca, que es possible adaptar el nou sistema amb els recursos materials que te un centre educatiu, sense deixar de banda de mirar de disminuir l'impacte de la petjada ecològica, social i econòmica per a que perjudiquin el mínim possible en nou entorn on s'ha instal·lat. Ho hem aconseguit molt bé els objectius que es preveia podrien ser els que més impacte tinguessin, com és el cas de la dotació d'infraestructura tecnològica. No sols s'ha aconseguit l'objectiu, sinó també tenir un efecte positiu, perquè s'ha allarga el cicle de vida tecnològic de la'parell i amb la virtualització de l'entorn, implica una reducció de cost econòmic, ambiental. I el desmantellament de la demostració va ser minim.

Capítol 10

Conclusions

10.1 Conclusions

Finalment arribem a les conclusions de la feina realitzada. Personalment ha sigut interessant treballar en un projecte de cooperació a nivell europeu, una oportunitat única per polir i desenvolupar eines de treball en grup i gestió personal.
Fins que finalment s'han acabat fent realitat els objectius proposats, has de conviure amb la pressió de complir amb els terminis d'entrega de la feina i la necessitat de saber adaptarte a diferents circumstàncies, maneres de fer i a qualsevol imprevist que hagi esdevingut. Això ha estat un dels aprenentatges, però l'impacte principal que m'emporto són tots els coneixements que he obtingut dins i fora de la carrera, que no podrien tenir millor posada en marxa en el món real.
L'aplicació dels continguts apresos en el projecte ha estat el més enriquidor de tot plegat, així com aprendre a resoldre les diferents dificultats que anaven apareixent.

10.1.1 Treball Assolit

El projecte ha culminat amb l'objectiu de crear l'entorn proposat inicialment, amb la mobilitat inclosa. Com hem pogut observar en el capítol 8, els temps estimats son d'entre 17 o 20 segons de salt entre NED i NED. Són temps relativament baixos, considerant tot l'entorn s'ha de regenerar en destí. Les proves fetes al centre d'ensenyament S'Arenal, en el municipi de Platja de Palma, van ser tot un èxit. Per tant creiem que pot posar-se en funcionament en qualsevol escenari, amb mínima penalització d'adaptació al nou entorn, un cost mínim, com s'ha esmentat en el capítol 9 i afegint el factor de seguretat.

10.1.2 Feina pel futur

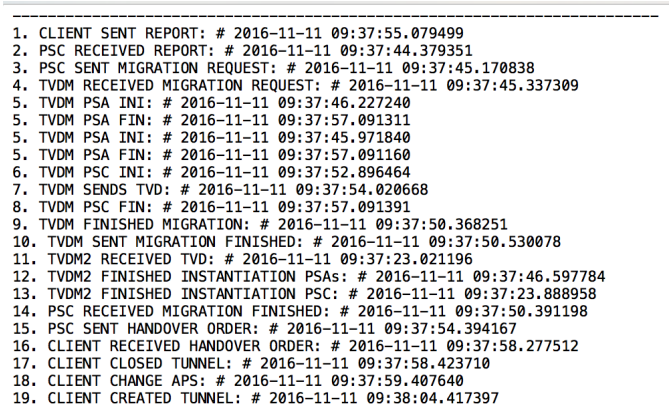
El projecte ha finalitzat i molts dels objectius s'han complert. No obstant sempre es pot prendre el relleu de la feina realitzada i seguir des d'aquest punt. Per exemple una possible integració amb NED treballant a nivell d'integració de xarxes wifi amb les les de 3G/4G. Per donar la possibilitat de portabilitat total. Actualment com bé ja hem parlat anteriorment, estem limitats en entorns locals.
Una altre línia de treball seria passar l'entorn a docker, o tecnologia similar, per reduir encara mes la mida de les maquines virtuals. Actualment es migra l'estat de la maquina i el disc dur. Amb docker es reduiria cosiderablament el volum de dades a moure entre els NEDs, facilitant així la possible integració del salt a xarxes de telefonia mobil. En el projecte SECURED es va prendre la decisió de treballar amb kvm/qemu per la facilitat

d'adaptar l'arquitectura de SECURED i evitar més complexitats per limitació de temps.

Com veiem tenim dos grans horitzons possibles per seguir desenvolupant tecnologia en aquest ámbit; el projecte SECURED va finalitzar al desembre del 2016 però esperem que hi pugui haver oportunitats per continuar desenvolupant aquest entorn creat o les seves diferents millores.

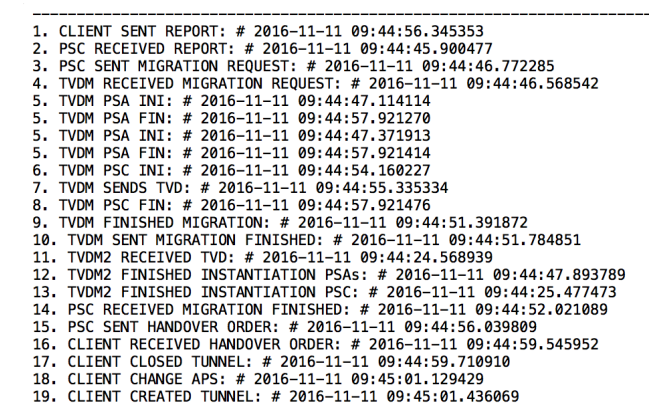
Apèndix A

Resultats de les proves



```
--------------------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-11 09:37:55.079499
2. PSC RECEIVED REPORT: # 2016-11-11 09:37:44.379351
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:37:45.170838
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:37:45.337309
5. TVDM PSA INI: # 2016-11-11 09:37:46.227240
5. TVDM PSA FIN: # 2016-11-11 09:37:57.091311
5. TVDM PSA INI: # 2016-11-11 09:37:45.971840
5. TVDM PSA FIN: # 2016-11-11 09:37:57.091160
6. TVDM PSC INI: # 2016-11-11 09:37:52.896464
7. TVDM SENDS TVD: # 2016-11-11 09:37:54.020668
8. TVDM PSC FIN: # 2016-11-11 09:37:57.091391
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:37:50.368251
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:37:50.530078
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:37:23.021196
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:37:46.597784
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:37:23.888958
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:37:50.391198
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:37:54.394167
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:37:58.277512
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:37:58.423710
18. CLIENT CHANGE APS: # 2016-11-11 09:37:59.407640
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:38:04.417397
```

Figura A.1: Execució nº20 realitzada el 11-11-2016



```
--------------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-11 09:44:56.345353
2. PSC RECEIVED REPORT: # 2016-11-11 09:44:45.900477
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:44:46.772285
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:44:46.568542
5. TVDM PSA INI: # 2016-11-11 09:44:47.114114
5. TVDM PSA FIN: # 2016-11-11 09:44:57.921270
5. TVDM PSA INI: # 2016-11-11 09:44:47.371913
5. TVDM PSA FIN: # 2016-11-11 09:44:57.921414
6. TVDM PSC INI: # 2016-11-11 09:44:54.160227
7. TVDM SENDS TVD: # 2016-11-11 09:44:55.335334
8. TVDM PSC FIN: # 2016-11-11 09:44:57.921476
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:44:51.391872
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:44:51.784851
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:44:24.568939
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:44:47.893789
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:44:25.477473
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:44:52.021089
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:44:56.039809
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:44:59.545952
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:44:59.710910
18. CLIENT CHANGE APS: # 2016-11-11 09:45:01.129429
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:45:01.436069
```

Figura A.2: Execució nº21 realitzada el 11-11-2016

```
-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-11 09:48:45.479626
2. PSC RECEIVED REPORT: # 2016-11-11 09:48:34.923037
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:48:35.792516
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:48:35.738216
5. TVDM PSA INI: # 2016-11-11 09:48:36.616042
5. TVDM PSA FIN: # 2016-11-11 09:48:47.894640
5. TVDM PSA INI: # 2016-11-11 09:48:36.344864
5. TVDM PSA FIN: # 2016-11-11 09:48:47.894500
6. TVDM PSC INI: # 2016-11-11 09:48:44.232937
7. TVDM SENDS TVD: # 2016-11-11 09:48:45.399733
8. TVDM PSC FIN: # 2016-11-11 09:48:47.894701
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:48:40.862358
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:48:41.869829
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:48:13.750302
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:48:37.940763
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:48:14.668532
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:48:41.961169
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:48:44.964971
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:48:48.641613
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:48:48.671886
18. CLIENT CHANGE APS: # 2016-11-11 09:48:54.209389
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:48:54.524096
```

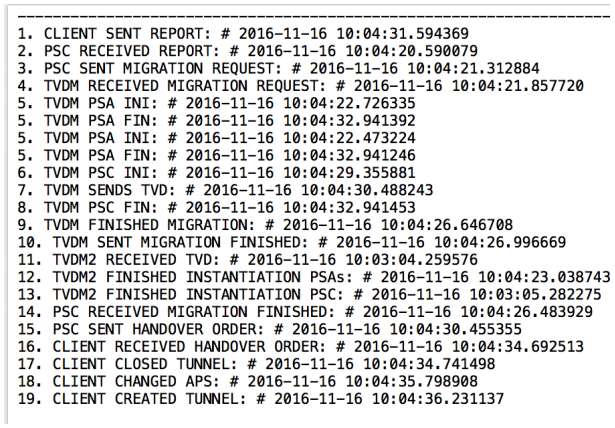
Figura A.3: Execució nº22 realitzada el 11-11-2016

```
-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-11 09:53:58.586584
2. PSC RECEIVED REPORT: # 2016-11-11 09:53:47.928598
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:53:48.623895
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:53:48.837392
5. TVDM PSA INI: # 2016-11-11 09:53:49.443251
5. TVDM PSA FIN: # 2016-11-11 09:54:00.006487
5. TVDM PSA INI: # 2016-11-11 09:53:49.720539
5. TVDM PSA FIN: # 2016-11-11 09:54:00.006641
6. TVDM PSC INI: # 2016-11-11 09:53:56.303038
7. TVDM SENDS TVD: # 2016-11-11 09:53:57.479060
8. TVDM PSC FIN: # 2016-11-11 09:54:00.006711
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:53:53.845887
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:53:53.949361
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:53:26.928860
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:53:50.015789
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:53:27.947198
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:53:53.760606
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:53:57.742009
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:54:01.651098
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:54:01.790274
18. CLIENT CHANGE APS: # 2016-11-11 09:54:07.182110
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:54:07.593441
```

Figura A.4: Execució nº23 realitzada el 11-11-2016

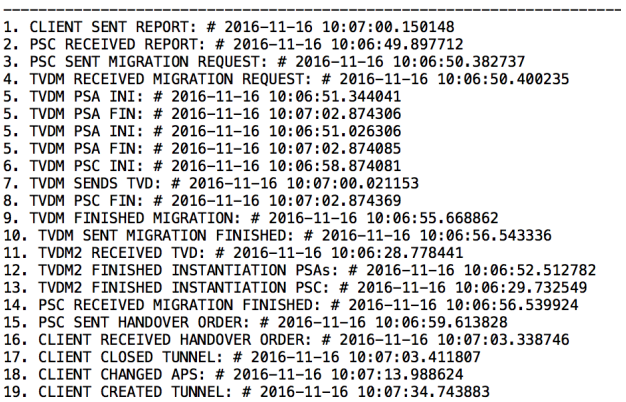
```
-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-11 09:58:29.524609
2. PSC RECEIVED REPORT: # 2016-11-11 09:58:18.991020
3. PSC SENT MIGRATION REQUEST: # 2016-11-11 09:58:19.697219
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-11 09:58:19.794643
5. TVDM PSA INI: # 2016-11-11 09:58:20.380155
5. TVDM PSA FIN: # 2016-11-11 09:58:31.053371
5. TVDM PSA INI: # 2016-11-11 09:58:20.634545
5. TVDM PSA FIN: # 2016-11-11 09:58:31.053431
6. TVDM PSC INI: # 2016-11-11 09:58:27.315045
7. TVDM SENDS TVD: # 2016-11-11 09:58:28.491057
8. TVDM PSC FIN: # 2016-11-11 09:58:31.053453
9. TVDM FINISHED MIGRATION: # 2016-11-11 09:58:24.795600
10. TVDM SENT MIGRATION FINISHED: # 2016-11-11 09:58:24.935476
11. TVDM2 RECEIVED TVD: # 2016-11-11 09:57:57.957200
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-11 09:58:21.034799
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-11 09:57:58.870283
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-11 09:58:24.867882
15. PSC SENT HANDOVER ORDER: # 2016-11-11 09:58:28.801807
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-11 09:58:32.595007
17. CLIENT CLOSED TUNNEL: # 2016-11-11 09:58:32.630394
18. CLIENT CHANGED APS: # 2016-11-11 09:58:34.111017
19. CLIENT CREATED TUNNEL: # 2016-11-11 09:58:34.433250
```

Figura A.5: Execució nº24 realitzada el 11-11-2016



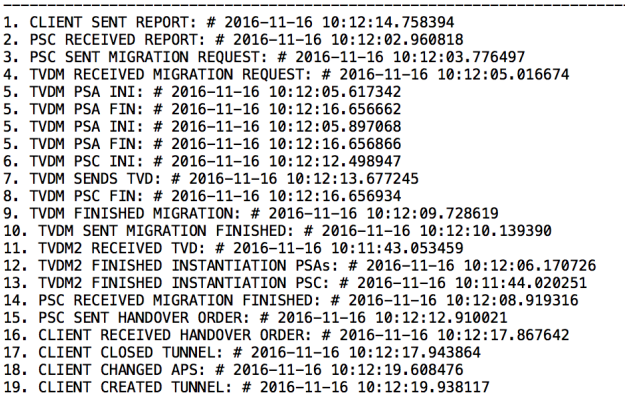
```
1. CLIENT SENT REPORT: # 2016-11-16 10:04:31.594369
2. PSC RECEIVED REPORT: # 2016-11-16 10:04:20.590079
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 10:04:21.312884
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 10:04:21.857720
5. TVDM PSA INI: # 2016-11-16 10:04:22.726335
5. TVDM PSA FIN: # 2016-11-16 10:04:32.941392
5. TVDM PSA INI: # 2016-11-16 10:04:22.473224
5. TVDM PSA FIN: # 2016-11-16 10:04:32.941246
6. TVDM PSC INI: # 2016-11-16 10:04:29.355881
7. TVDM SENDS TVD: # 2016-11-16 10:04:30.488243
8. TVDM PSC FIN: # 2016-11-16 10:04:32.941453
9. TVDM FINISHED MIGRATION: # 2016-11-16 10:04:26.646708
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 10:04:26.996669
11. TVDM2 RECEIVED TVD: # 2016-11-16 10:03:04.259576
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 10:04:23.038743
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 10:03:05.282275
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 10:04:26.483929
15. PSC SENT HANDOVER ORDER: # 2016-11-16 10:04:30.455355
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 10:04:34.692513
17. CLIENT CLOSED TUNNEL: # 2016-11-16 10:04:34.741498
18. CLIENT CHANGED APS: # 2016-11-16 10:04:35.798908
19. CLIENT CREATED TUNNEL: # 2016-11-16 10:04:36.231137
```

Figura A.6: Execució nº44 realitzada el 16-11-2016



```
1. CLIENT SENT REPORT: # 2016-11-16 10:07:00.150148
2. PSC RECEIVED REPORT: # 2016-11-16 10:06:49.897712
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 10:06:50.382737
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 10:06:50.400235
5. TVDM PSA INI: # 2016-11-16 10:06:51.344041
5. TVDM PSA FIN: # 2016-11-16 10:07:02.874306
5. TVDM PSA INI: # 2016-11-16 10:06:51.026306
5. TVDM PSA FIN: # 2016-11-16 10:07:02.874085
6. TVDM PSC INI: # 2016-11-16 10:06:58.874081
7. TVDM SENDS TVD: # 2016-11-16 10:07:00.021153
8. TVDM PSC FIN: # 2016-11-16 10:07:02.874369
9. TVDM FINISHED MIGRATION: # 2016-11-16 10:06:55.668862
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 10:06:56.543336
11. TVDM2 RECEIVED TVD: # 2016-11-16 10:06:28.778441
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 10:06:52.512782
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 10:06:29.732549
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 10:06:56.539924
15. PSC SENT HANDOVER ORDER: # 2016-11-16 10:06:59.613828
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 10:07:03.338746
17. CLIENT CLOSED TUNNEL: # 2016-11-16 10:07:03.411807
18. CLIENT CHANGED APS: # 2016-11-16 10:07:13.988624
19. CLIENT CREATED TUNNEL: # 2016-11-16 10:07:34.743883
```

Figura A.7: Execució nº45 realitzada el 16-11-2016



```
1. CLIENT SENT REPORT: # 2016-11-16 10:12:14.758394
2. PSC RECEIVED REPORT: # 2016-11-16 10:12:02.960818
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 10:12:03.776497
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 10:12:05.016674
5. TVDM PSA INI: # 2016-11-16 10:12:05.617342
5. TVDM PSA FIN: # 2016-11-16 10:12:16.656662
5. TVDM PSA INI: # 2016-11-16 10:12:05.897068
5. TVDM PSA FIN: # 2016-11-16 10:12:16.656866
6. TVDM PSC INI: # 2016-11-16 10:12:12.498947
7. TVDM SENDS TVD: # 2016-11-16 10:12:13.677245
8. TVDM PSC FIN: # 2016-11-16 10:12:16.656934
9. TVDM FINISHED MIGRATION: # 2016-11-16 10:12:09.728619
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 10:12:10.139390
11. TVDM2 RECEIVED TVD: # 2016-11-16 10:11:43.053459
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 10:12:06.170726
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 10:11:44.020251
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 10:12:08.919316
15. PSC SENT HANDOVER ORDER: # 2016-11-16 10:12:12.910021
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 10:12:17.867642
17. CLIENT CLOSED TUNNEL: # 2016-11-16 10:12:17.943864
18. CLIENT CHANGED APS: # 2016-11-16 10:12:19.608476
19. CLIENT CREATED TUNNEL: # 2016-11-16 10:12:19.938117
```

Figura A.8: Execució nº46 realitzada el 16-11-2016

```
-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-16 10:57:05.531005
2. PSC RECEIVED REPORT: # 2016-11-16 10:56:54.861177
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 10:56:55.520417
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 10:56:55.783313
5. TVDM PSA INI: # 2016-11-16 10:56:56.411813
5. TVDM PSA FIN: # 2016-11-16 10:57:08.069434
5. TVDM PSA INI: # 2016-11-16 10:56:56.689222
5. TVDM PSA FIN: # 2016-11-16 10:57:08.069542
6. TVDM PSC INI: # 2016-11-16 10:57:04.370320
7. TVDM SENDS TVD: # 2016-11-16 10:57:05.538566
8. TVDM PSC FIN: # 2016-11-16 10:57:08.069582
9. TVDM FINISHED MIGRATION: # 2016-11-16 10:57:00.960346
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 10:57:01.974575
11. TVDM2 RECEIVED TVD: # 2016-11-16 10:56:33.975503
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 10:56:57.971758
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 10:56:34.948030
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 10:57:01.732680
15. PSC SENT HANDOVER ORDER: # 2016-11-16 10:57:04.616133
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 10:57:08.595589
17. CLIENT CLOSED TUNNEL: # 2016-11-16 10:57:08.638575
18. CLIENT CHANGED APS: # 2016-11-16 10:57:14.231371
19. CLIENT CREATED TUNNEL: # 2016-11-16 10:57:14.775897
```

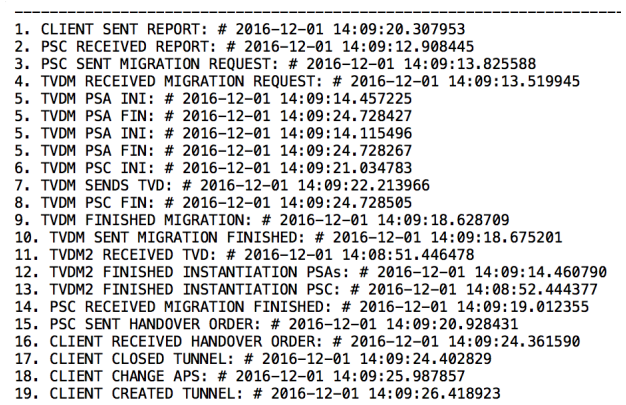
Figura A.9: Execució nº47 realitzada el 16-11-2016

```
├-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-11-16 13:50:13.645237
2. PSC RECEIVED REPORT: # 2016-11-16 13:50:02.932151
3. PSC SENT MIGRATION REQUEST: # 2016-11-16 13:50:03.639777
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-11-16 13:50:03.863320
5. TVDM PSA INI: # 2016-11-16 13:50:04.526736
5. TVDM PSA FIN: # 2016-11-16 13:50:15.496560
5. TVDM PSA INI: # 2016-11-16 13:50:04.778907
5. TVDM PSA FIN: # 2016-11-16 13:50:15.496688
6. TVDM PSC INI: # 2016-11-16 13:50:11.386086
7. TVDM SENDS TVD: # 2016-11-16 13:50:12.577279
8. TVDM PSC FIN: # 2016-11-16 13:50:15.496738
9. TVDM FINISHED MIGRATION: # 2016-11-16 13:50:08.938921
10. TVDM SENT MIGRATION FINISHED: # 2016-11-16 13:50:09.033690
11. TVDM2 RECEIVED TVD: # 2016-11-16 13:49:41.996291
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-11-16 13:50:04.787727
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-11-16 13:49:42.967178
14. PSC RECEIVED MIGRATION FINISHED: # 2016-11-16 13:50:08.851430
15. PSC SENT HANDOVER ORDER: # 2016-11-16 13:50:12.871162
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-11-16 13:50:16.824559
17. CLIENT CLOSED TUNNEL: # 2016-11-16 13:50:16.868726
18. CLIENT CHANGED APS: # 2016-11-16 13:50:22.432739
19. CLIENT CREATED TUNNEL: # 2016-11-16 13:50:22.988181
```

Figura A.10: Execució nº48 realitzada el 16-11-2016

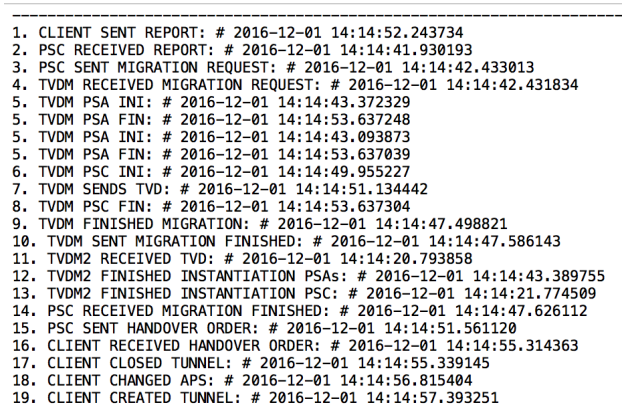
```
-----------------------------------------------------------------
1. CLIENT SENT REPORT: # 2016-12-01 13:40:59.603435
2. PSC RECEIVED REPORT: # 2016-12-01 13:40:49.191078
3. PSC SENT MIGRATION REQUEST: # 2016-12-01 13:40:49.977448
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-01 13:40:49.833042
5. TVDM PSA INI: # 2016-12-01 13:40:50.746877
5. TVDM PSA FIN: # 2016-12-01 13:41:00.932879
5. TVDM PSA INI: # 2016-12-01 13:40:50.459558
5. TVDM PSA FIN: # 2016-12-01 13:41:00.932739
6. TVDM PSC INI: # 2016-12-01 13:40:57.287511
7. TVDM SENDS TVD: # 2016-12-01 13:40:58.463298
8. TVDM PSC FIN: # 2016-12-01 13:41:00.932945
9. TVDM FINISHED MIGRATION: # 2016-12-01 13:40:54.428560
10. TVDM SENT MIGRATION FINISHED: # 2016-12-01 13:40:54.947463
11. TVDM2 RECEIVED TVD: # 2016-12-01 13:40:17.662428
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-01 13:40:50.734782
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-01 13:40:18.569856
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-01 13:40:55.107131
15. PSC SENT HANDOVER ORDER: # 2016-12-01 13:40:59.126046
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-01 13:41:02.724006
17. CLIENT CLOSED TUNNEL: # 2016-12-01 13:41:02.761684
18. CLIENT CHANGED APS: # 2016-12-01 13:41:04.227252
19. CLIENT CREATED TUNNEL: # 2016-12-01 13:41:04.706947
```

Figura A.11: Execució nº35 realitzada el 01-12-2016



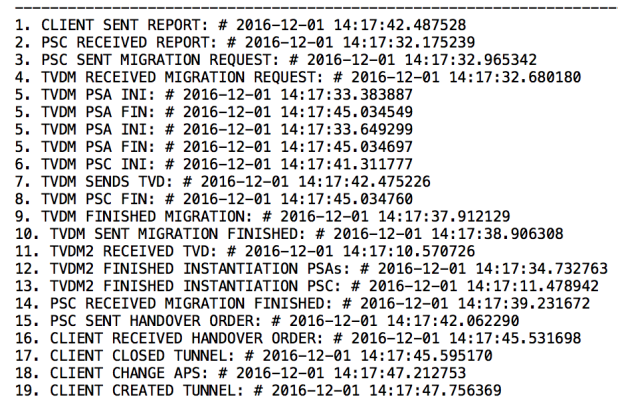
```
1. CLIENT SENT REPORT: # 2016-12-01 14:09:20.307953
2. PSC RECEIVED REPORT: # 2016-12-01 14:09:12.908445
3. PSC SENT MIGRATION REQUEST: # 2016-12-01 14:09:13.825588
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-01 14:09:13.519945
5. TVDM PSA INI: # 2016-12-01 14:09:14.457225
5. TVDM PSA FIN: # 2016-12-01 14:09:24.728427
5. TVDM PSA INI: # 2016-12-01 14:09:14.115496
5. TVDM PSA FIN: # 2016-12-01 14:09:24.728267
6. TVDM PSC INI: # 2016-12-01 14:09:21.034783
7. TVDM SENDS TVD: # 2016-12-01 14:09:22.213966
8. TVDM PSC FIN: # 2016-12-01 14:09:24.728505
9. TVDM FINISHED MIGRATION: # 2016-12-01 14:09:18.628709
10. TVDM SENT MIGRATION FINISHED: # 2016-12-01 14:09:18.675201
11. TVDM2 RECEIVED TVD: # 2016-12-01 14:08:51.446478
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-01 14:09:14.460790
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-01 14:08:52.444377
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-01 14:09:19.012355
15. PSC SENT HANDOVER ORDER: # 2016-12-01 14:09:20.928431
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-01 14:09:24.361590
17. CLIENT CLOSED TUNNEL: # 2016-12-01 14:09:24.402829
18. CLIENT CHANGE APS: # 2016-12-01 14:09:25.987857
19. CLIENT CREATED TUNNEL: # 2016-12-01 14:09:26.418923
```

Figura A.12: Execució nº36 realitzada el 01-12-2016



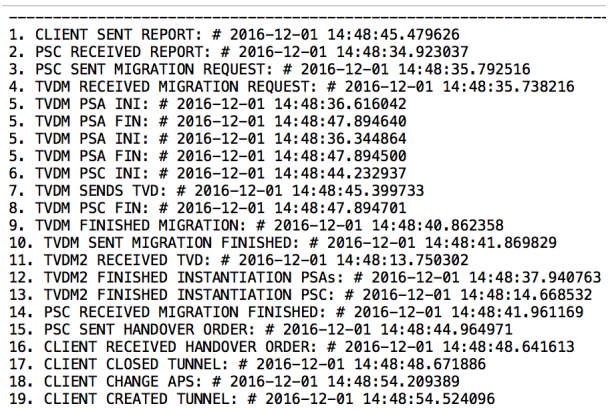
```
1. CLIENT SENT REPORT: # 2016-12-01 14:14:52.243734
2. PSC RECEIVED REPORT: # 2016-12-01 14:14:41.930193
3. PSC SENT MIGRATION REQUEST: # 2016-12-01 14:14:42.433013
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-01 14:14:42.431834
5. TVDM PSA INI: # 2016-12-01 14:14:43.372329
5. TVDM PSA FIN: # 2016-12-01 14:14:53.637248
5. TVDM PSA INI: # 2016-12-01 14:14:43.093873
5. TVDM PSA FIN: # 2016-12-01 14:14:53.637039
6. TVDM PSC INI: # 2016-12-01 14:14:49.955227
7. TVDM SENDS TVD: # 2016-12-01 14:14:51.134442
8. TVDM PSC FIN: # 2016-12-01 14:14:53.637304
9. TVDM FINISHED MIGRATION: # 2016-12-01 14:14:47.498821
10. TVDM SENT MIGRATION FINISHED: # 2016-12-01 14:14:47.586143
11. TVDM2 RECEIVED TVD: # 2016-12-01 14:14:20.793858
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-01 14:14:43.389755
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-01 14:14:21.774509
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-01 14:14:47.626112
15. PSC SENT HANDOVER ORDER: # 2016-12-01 14:14:51.561120
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-01 14:14:55.314363
17. CLIENT CLOSED TUNNEL: # 2016-12-01 14:14:55.339145
18. CLIENT CHANGED APS: # 2016-12-01 14:14:56.815404
19. CLIENT CREATED TUNNEL: # 2016-12-01 14:14:57.393251
```

Figura A.13: Execució nº37 realitzada el 01-12-2016



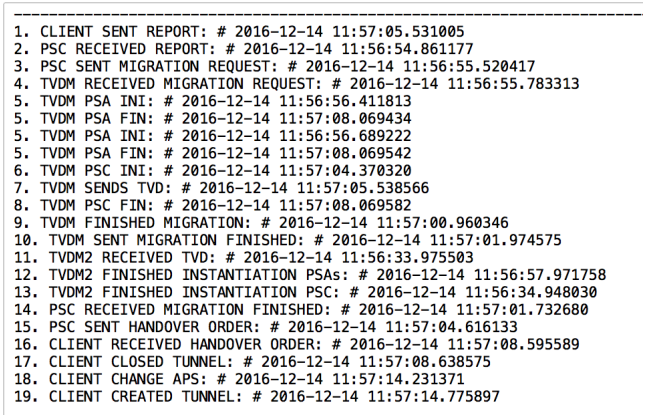
```
1. CLIENT SENT REPORT: # 2016-12-01 14:17:42.487528
2. PSC RECEIVED REPORT: # 2016-12-01 14:17:32.175239
3. PSC SENT MIGRATION REQUEST: # 2016-12-01 14:17:32.965342
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-01 14:17:32.680180
5. TVDM PSA INI: # 2016-12-01 14:17:33.383887
5. TVDM PSA FIN: # 2016-12-01 14:17:45.034549
5. TVDM PSA INI: # 2016-12-01 14:17:33.649299
5. TVDM PSA FIN: # 2016-12-01 14:17:45.034697
6. TVDM PSC INI: # 2016-12-01 14:17:41.311777
7. TVDM SENDS TVD: # 2016-12-01 14:17:42.475226
8. TVDM PSC FIN: # 2016-12-01 14:17:45.034760
9. TVDM FINISHED MIGRATION: # 2016-12-01 14:17:37.912129
10. TVDM SENT MIGRATION FINISHED: # 2016-12-01 14:17:38.906308
11. TVDM2 RECEIVED TVD: # 2016-12-01 14:17:10.570726
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-01 14:17:34.732763
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-01 14:17:11.478942
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-01 14:17:39.231672
15. PSC SENT HANDOVER ORDER: # 2016-12-01 14:17:42.062290
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-01 14:17:45.531698
17. CLIENT CLOSED TUNNEL: # 2016-12-01 14:17:45.595170
18. CLIENT CHANGE APS: # 2016-12-01 14:17:47.212753
19. CLIENT CREATED TUNNEL: # 2016-12-01 14:17:47.756369
```

Figura A.14: Execució nº38 realitzada el 16-11-2016



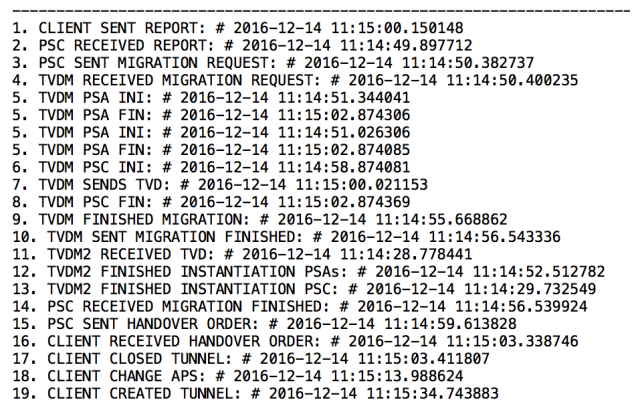
```
1. CLIENT SENT REPORT: # 2016-12-01 14:48:45.479626
2. PSC RECEIVED REPORT: # 2016-12-01 14:48:34.923037
3. PSC SENT MIGRATION REQUEST: # 2016-12-01 14:48:35.792516
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-01 14:48:35.738216
5. TVDM PSA INI: # 2016-12-01 14:48:36.616042
5. TVDM PSA FIN: # 2016-12-01 14:48:47.894640
5. TVDM PSA INI: # 2016-12-01 14:48:36.344864
5. TVDM PSA FIN: # 2016-12-01 14:48:47.894500
6. TVDM PSC INI: # 2016-12-01 14:48:44.232937
7. TVDM SENDS TVD: # 2016-12-01 14:48:45.399733
8. TVDM PSC FIN: # 2016-12-01 14:48:47.894701
9. TVDM FINISHED MIGRATION: # 2016-12-01 14:48:40.862358
10. TVDM SENT MIGRATION FINISHED: # 2016-12-01 14:48:41.869829
11. TVDM2 RECEIVED TVD: # 2016-12-01 14:48:13.750302
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-01 14:48:37.940763
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-01 14:48:14.668532
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-01 14:48:41.961169
15. PSC SENT HANDOVER ORDER: # 2016-12-01 14:48:44.964971
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-01 14:48:48.641613
17. CLIENT CLOSED TUNNEL: # 2016-12-01 14:48:48.671886
18. CLIENT CHANGE APS: # 2016-12-01 14:48:54.209389
19. CLIENT CREATED TUNNEL: # 2016-12-01 14:48:54.524096
```

Figura A.15: Execució nº39 realitzada el 01-12-2016



```
1. CLIENT SENT REPORT: # 2016-12-14 11:57:05.531005
2. PSC RECEIVED REPORT: # 2016-12-14 11:56:54.861177
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:56:55.520417
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:56:55.783313
5. TVDM PSA INI: # 2016-12-14 11:56:56.411813
5. TVDM PSA FIN: # 2016-12-14 11:57:08.069434
5. TVDM PSA INI: # 2016-12-14 11:56:56.689222
5. TVDM PSA FIN: # 2016-12-14 11:57:08.069542
6. TVDM PSC INI: # 2016-12-14 11:57:04.370320
7. TVDM SENDS TVD: # 2016-12-14 11:57:05.538566
8. TVDM PSC FIN: # 2016-12-14 11:57:08.069582
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:57:00.960346
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:57:01.974575
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:56:33.975503
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:56:57.971758
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:56:34.948030
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:57:01.732680
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:57:04.616133
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:57:08.595589
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:57:08.638575
18. CLIENT CHANGE APS: # 2016-12-14 11:57:14.231371
19. CLIENT CREATED TUNNEL: # 2016-12-14 11:57:14.775897
```

Figura A.16: Execució nº26 realitzada el 14-12-2016



```
1. CLIENT SENT REPORT: # 2016-12-14 11:15:00.150148
2. PSC RECEIVED REPORT: # 2016-12-14 11:14:49.897712
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:14:50.382737
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:14:50.400235
5. TVDM PSA INI: # 2016-12-14 11:14:51.344041
5. TVDM PSA FIN: # 2016-12-14 11:15:02.874306
5. TVDM PSA INI: # 2016-12-14 11:14:51.026306
5. TVDM PSA FIN: # 2016-12-14 11:15:02.874085
6. TVDM PSC INI: # 2016-12-14 11:14:58.874081
7. TVDM SENDS TVD: # 2016-12-14 11:15:00.021153
8. TVDM PSC FIN: # 2016-12-14 11:15:02.874369
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:14:55.668862
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:14:56.543336
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:14:28.778441
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:14:52.512782
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:14:29.732549
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:14:56.539924
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:14:59.613828
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:15:03.338746
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:15:03.411807
18. CLIENT CHANGE APS: # 2016-12-14 11:15:13.988624
19. CLIENT CREATED TUNNEL: # 2016-12-14 11:15:34.743883
```

Figura A.17: Execució nº27 realitzada el 14-12-2016

```
1. CLIENT SENT REPORT: # 2016-12-14 11:22:59.603435
2. PSC RECEIVED REPORT: # 2016-12-14 11:22:49.191078
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:22:49.977448
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:22:49.833042
5. TVDM PSA INI: # 2016-12-14 11:22:50.746877
5. TVDM PSA FIN: # 2016-12-14 11:23:00.932879
5. TVDM PSA INI: # 2016-12-14 11:22:50.459558
5. TVDM PSA FIN: # 2016-12-14 11:23:00.932739
6. TVDM PSC INI: # 2016-12-14 11:22:57.287511
7. TVDM SENDS TVD: # 2016-12-14 11:22:58.463298
8. TVDM PSC FIN: # 2016-12-14 11:23:00.932945
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:22:54.428560
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:22:54.947463
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:22:17.662428
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:22:50.734782
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:22:18.569856
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:22:55.107131
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:22:59.126046
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:23:02.724006
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:23:02.761684
18. CLIENT CHANGE APS: # 2016-12-14 11:23:04.227252
19. CLIENT CREATED TUNNEL: # 2016-12-14 11:23:04.706947
```

Figura A.18: Execució nº28 realitzada el 14-12-2016

```
1. CLIENT SENT REPORT: # 2016-12-14 11:33:55.079499
2. PSC RECEIVED REPORT: # 2016-12-14 11:33:44.379351
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:33:45.170838
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:33:45.337309
5. TVDM PSA INI: # 2016-12-14 11:33:46.227240
5. TVDM PSA FIN: # 2016-12-14 11:33:57.091311
5. TVDM PSA INI: # 2016-12-14 11:33:45.971840
5. TVDM PSA FIN: # 2016-12-14 11:33:57.091160
6. TVDM PSC INI: # 2016-12-14 11:33:52.896464
7. TVDM SENDS TVD: # 2016-12-14 11:33:54.020668
8. TVDM PSC FIN: # 2016-12-14 11:33:57.091391
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:33:50.368251
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:33:50.530078
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:33:23.021196
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:33:46.597784
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:33:23.888958
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:33:50.391198
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:33:54.394167
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:33:58.277512
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:33:58.423710
18. CLIENT CHANGE APS: # 2016-12-14 11:33:59.407640
19. CLIENT CREATED TUNNEL: # 2016-12-14 09:38:04.417397
```

Figura A.19: Execució nº29 realitzada el 14-12-2016

```
1. CLIENT SENT REPORT: # 2016-12-14 11:55:45.479626
2. PSC RECEIVED REPORT: # 2016-12-14 11:55:34.923037
3. PSC SENT MIGRATION REQUEST: # 2016-12-14 11:55:35.792516
4. TVDM RECEIVED MIGRATION REQUEST: # 2016-12-14 11:55:35.738216
5. TVDM PSA INI: # 2016-12-14 11:55:36.616042
5. TVDM PSA FIN: # 2016-12-14 11:55:47.894640
5. TVDM PSA INI: # 2016-12-14 11:55:36.344864
5. TVDM PSA FIN: # 2016-12-14 11:55:47.894500
6. TVDM PSC INI: # 2016-12-14 11:55:44.232937
7. TVDM SENDS TVD: # 2016-12-14 11:55:45.399733
8. TVDM PSC FIN: # 2016-12-14 11:55:47.894701
9. TVDM FINISHED MIGRATION: # 2016-12-14 11:55:40.862358
10. TVDM SENT MIGRATION FINISHED: # 2016-12-14 11:55:41.869829
11. TVDM2 RECEIVED TVD: # 2016-12-14 11:55:13.750302
12. TVDM2 FINISHED INSTANTIATION PSAs: # 2016-12-14 11:55:37.940763
13. TVDM2 FINISHED INSTANTIATION PSC: # 2016-12-14 11:55:14.668532
14. PSC RECEIVED MIGRATION FINISHED: # 2016-12-14 11:55:41.961169
15. PSC SENT HANDOVER ORDER: # 2016-12-14 11:55:44.964971
16. CLIENT RECEIVED HANDOVER ORDER: # 2016-12-14 11:55:48.641613
17. CLIENT CLOSED TUNNEL: # 2016-12-14 11:55:48.671886
18. CLIENT CHANGE APS: # 2016-12-14 11:55:54.209389
19. CLIENT CREATED TUNNEL: # 2016-12-14 11:55:54.524096
```

Figura A.20: Execució nº48 realitzada el 14-12-2016

Apèndix B

Codis del projecte

B.1 Codis amb funcionalitats afegides

Fragment de codi B.1: mainIPSEC.py

```
'''
Created on 23/mag/2014

@author: rbonafiglia

Main script to launch from gunicorn to star the orchestrator WSGI server
'''
import falcon
import Config
from Orchestrator import Orchestrator
from GraphInstatiator import GraphInstantiator
from PSAcreation import PSAcreation
from GraphInfo import GraphInfo
from PSAConf import PSAConf
###migration code###
from TVDMigration import TVDMigration
###
from VerifierCache import VerifierCache
import logging
import datetime as dt
import signal
import sys

# TODO: control if there are sessions open, if true close them
class MyFormatter(logging.Formatter):
    converter=dt.datetime.fromtimestamp
    def formatTime(self, record, datefmt=None):
        ct = self.converter(record.created)
        if datefmt:
            s = ct.strftime(datefmt)
        else:
            t = ct.strftime("%Y-%m-%d %H:%M:%S")
            s = "%s,%03d" % (t, record.msecs)
        return s


```

```
conf = Config.Configuration()
#logging.config.fileConfig(conf.LOG_FILE)
logger = logging.getLogger(__name__)
logger.setLevel(logging.DEBUG)

fh = logging.FileHandler(conf.LOG_FILE)
fh.setLevel(logging.DEBUG)

console = logging.StreamHandler()

formatter = MyFormatter(fmt='%(asctime)s %(message)s',datefmt='%Y-%m-%d,%H:%M:%S.%f')
fh.setFormatter(formatter)
console.setFormatter(formatter)

logger.addHandler(console)
logger.addHandler(fh)
#logging.basicConfig(filename=conf.LOG_FILE,level=logging.DEBUG, format='%(asctime)s %(message)s', datefmt='%m/%d/%Y %I:%M:%S %p')
logger.info("--------")
logger.info("NED / TVDM init.")
logger.info("NED / TVDM VERSION: " + str(conf.TVDM_VERSION))
logger.info("--------")
# Falcon starts

app = falcon.API()
instantiator = GraphInstantiator(conf, logger, True)
###start migration code###
migration = TVDMigration(instantiator, conf, logger)
###end migration code###
def signal_term_handler(signal, frame):
    logger.info("SIG TERM")
    if instantiator.signal_term_handler():
        logger.info("FINISH DESTROY ALL TVD")
            sys.exit(0)

signal.signal(signal.SIGTERM, signal_term_handler)
###migration code -> added parameter migration###
orch = Orchestrator(instantiator, migration)
###

### Verifier cache
vc = VerifierCache(conf)
vc.start()
app.add_route('/verify', vc)

app.add_route('/instantiateTVD', orch)
###migration code -> added new call###
app.add_route('/migration', orch)
###
psa = PSAcreation(instantiator)
app.add_route('/createPSA', psa)

graph = GraphInfo(instantiator,conf.USER_GRAPH_LOCATION, conf)
app.add_route('/getGraph', graph)
```

```
90
91 psaConfRes = PSAConf(conf.PSA_CONF_LOCATION, logger, conf,
       instantiator)
92 app.add_route('/getConf/{psa_id}/{conf_id}', psaConfRes)
```

Fragment de codi B.2: Orchstrator.py

```
1  import falcon
2  import json
3  from threading import Thread
4  from threading import Event
5  import thread
6  from GraphInstatiator import GraphInstantiator
7  import logging
8  import sys
9  import subprocess
10 import time
11 from datetime import datetime
12
13
14 class Orchestrator(object):
15     '''
16     Orchestrator class that intercept the REST call through the
           WSGI server
17     '''
18
19     def __init__(self, instantiator, TVDMigration):
20         '''
21         Constructor
22         '''
23         self.instantiator = instantiator
24         self.TVDMigration = TVDMigration
25         self.migrate = False
26         self.eventList = {}
27         self.handoverEvents = {}
28         self.obj = {}
29         self.obj2 = {}
30
31     def on_delete(self, request, response):
32         try:
33             args = request.stream.read()
34             self.instantiator.logger.info(request.method + " " +
                   request.uri + " " + args)
35             session = json.loads(args, 'utf-8')
36             token = self.instantiator.IPandUser[session["IP"]]
37             newTVD = Thread(target=self.instantiator.deleteUser,
                   kwargs={"session": session})
38             newTVD.start()
39             response.status = falcon.HTTP_200
40         except Exception as e:
41             self.instantiator.logger.exception(sys.exc_info()[0])
42             response.status = falcon.HTTP_501
43
44     ######start migration code############
45
46     def on_put(self, request, response):
```

```
47          '''
48          shared by the instantiation and migration of TVD
49          '''
50          try:
51              self.instantiator.timestamps["tvdm2_recieves_TVD"] =
                    datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
52              args = request.stream.read()
53              session = json.loads(args, 'utf-8')
54              self.instantiator.logger.info(request.method + " " +
                    request.uri + " " + json.dumps(session, indent=4,
                    sort_keys=True))
55              if "action" in session:
56                  if session["action"] == "TVD":
57                      newTVD = Thread(target=self.instantiator.
                            instatiateTVD, kwargs={"session": session
                            })
58                      newTVD.start()
59                      response.status = falcon.HTTP_200
60                  else:
61                      self.instantiator.timestamps = session["
                            timestamps"]
62                      response.status = falcon.HTTP_200
63              else:
64                  newTVD = Thread(target=self.instantiator.
                        instatiateTVD, kwargs={"session": session})
65                  newTVD.start()
66                  response.status = falcon.HTTP_200
67          except Exception as e:
68              self.instantiator.logger.exception(sys.exc_info()[0])
69              response.status = falcon.HTTP_501
70
71      def on_post(self, request, response):
72          '''
73          exclusive for migration
74          '''
75          try:
76              self.instantiator.tvdm_receives_migration_request =
                    datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
77              args = request.stream.read()
78              self.TVDMigration.instantiator.logger.info(request.
                    method + " " + request.uri + " " + args)
79              session = json.loads(args, 'utf-8')
80
81              self.eventList[session["token"]] = Event()
82              self.handoverEvents[session["token"]] = Event()
83              migration = Thread(name=session["token"], target=self.
                    TVDMigration.init_migration,
84                                 args=(self.eventList[session["token
                                        "]], self.handoverEvents[
                                        session["token"]],), kwargs={"
                                        session": session})
85              migration.start()
86              response.status = falcon.HTTP_200
87          except:
88              self.TVDMigration.instantiator.logger.exception(sys.
                    exc_info()[0])
```

```
89              response.status = falcon.HTTP_500
90
91      def get_client_address(self, environ):
92
93          try:
94
95              return environ['HTTP_X_FORWARDED_FOR'].split(',')[-1].
                    strip()
96
97          except KeyError:
98
99              return environ['REMOTE_ADDR']
100
101
102
103     def on_get(self, request, response):
104         '''
105         Exclusive for migration, non-bloking query migration
106         '''
107         try:
108             self.TVDMigration.instantiator.logger.info(request.
                    method + " " + request.uri)
109             user = request.get_param("user")
110             action = request.get_param("action")
111
112             obj = {}
113             if action == "migration":
114                 if user in self.eventList.keys():
115                     self.TVDMigration.instantiator.logger.info("
                            user: %s, eventList: %s" % (str(user), str
                            (self.eventList[user].isSet())))
116                     obj = {"status": self.eventList[user].isSet()}
117                     if self.eventList[user].isSet() is True:
118                         self.instantiator.
                                tvdm_sends_migration_finished =
                                datetime.utcnow().strftime("%Y-%m-%d %
                                H:%M:%S.%f")
119                         self.handoverEvents[user].set()
120                     else:
121                         obj = {"status": False}
122             else:
123                 obj["timestamps"] = self.instantiator.timestamps
124             response.body = json.dumps(obj)
125
126             self.instantiator.logger.info(str(response.body))
127             response.status = falcon.HTTP_200
128
129         except Exception as e:
130             self.instantiator.logger.exception(sys.exc_info()[0])
131             self.instantiator.logger.exception(str(e))
132             response.status = falcon.HTTP_501
133     #######end migration code###############
```

Fragment de codi B.3: GraphInstatiator.py

```
1 '''
```

```
Created on 29/lug/2014

'''
import json
import subprocess, shlex
from Compute import Compute
from Network import Network
from userTVD import UserTVD
from userTVDIPSECLess import UserTVD as UserTVDIPSECLess
import requests
import ast
import commands
import logging
from datetime import datetime

class GraphInstantiator(object):
    '''
    Class used to instantiate the TVD and the Profile Graph
    '''

    def __init__(self, configure, logger, useIPSEC=False):
        '''
        Constructor
        '''

        self.client_ip = None
        subprocess.call(["bash", configure.SCRIPT_LOCATION + "
            emptyDHCP.sh"])
        self.useIPSEC = useIPSEC
        self.IPandUser = {}  # Used to associate the token to the
            IP (for the logoutelf.IPandUser = {}  # Used to
            associate the token to the IP (for the logout
        self.logger = logger
        self.config = configure
        self.networkManager = Network(configure)
        self.computeManager = Compute(configure)
        self.userTVDs = {}  # All the generated TVD
        self.TokenIP = {}  # Associate the PSC IP to a specifc
            user
        self.sigTerm = False
        self.migration = False #flag migration
        self.psa_ip_route_table = int(configure.PSA_IP_ROUTE_TABLE
            )
        self.default_ssid = configure.DEFAULT_SSID
        self.local_info = configure.migration_ned_info(self.
            default_ssid)
        if type(self.local_info) is str:
            self.local_info = ast.literal_eval(self.local_info)
        self.local_host = self.local_info['ned_ip']
        self.local_port = int(self.local_info['ned_port'])
        self.defaultnamespace = configure.DEFAULT_NAME_SPACE
        self.default_ssid = configure.DEFAULT_SSID
        self.mobility = configure.MOBILITY
        self.ip_ext = configure.IP_EXT
        self.gw_ip = self.config.GATEWAY_IP
        #####test#####
```

```
52          self.tvdm_sends_migration_finished  = None
53          self.tvdm_receives_migration_request = None
54          self.psas_tt = None
55          self.tvdm_ini_psc = None
56          self.tvdm_sends_TVD = None
57          self.tvdm_fin_psc = None
58          self.tvdm_finishes_migration = None
59          self.timestamps = {}
60
61
62
63
64      def instatiateTVD(self, session):
65          '''
66          Intantiate the TVD for a logged user.
67          In case for a TVD already instatiated it will generate the
                flows used for reedirecting the traffic of the
68          new defice on it
69          '''
70          self.client_ip = str(session['IP'])
71
72
73          if 'migration' in session:
74              self.migration = bool(session['migration'])
75              self.logger.info("INSTANTIATE TVD WITH migration=%s" %
                    (str(session['migration'])))
76
77          ###start migration code###
78          if self.migration:
79              self.logger.info("migration TVD " + str(session['IP'])
                    )
80          else:
81              self.logger.info("instantiation TVD")
82          ###end migration code###
83
84          self.logger.info("IP " + session['IP'])
85          if not self.migration:
86              if session['IP'] in self.IPandUser.keys():
87                  self.logger.info("Machine already logged")
88                  return
89
90          self.IPandUser[session['IP']] = session['token']
91
92          if session['token'] in self.userTVDs.keys():
93              self.logger.info("psaLIST %s" % (str(session['PSAs']))
                    )
94
95              userTVD = self.userTVDs[session['token']]
96              if self.migration:
97                  userTVD.migration = self.migration
98
99              if userTVD.migration:
100                 self.logger.info("userTVD migration %s" % (str(
                        userTVD.migration)))
101             userTVD.addNewIP(str(session['IP']))
102             userTVD.generatePSCflows()
```

```
103             self.instantiatePSA(session)
104             return
105 
106         ###start migraton Code###
107          if self.migration:  # If the user TVD migrated
108             vlanID = session['vlanID']
109          else:
110          ###end migration code###
111             # If the user have no TVD instantiated
112             vlanID = self.networkManager.getNewVLANID()
113 
114          ###start migraton Code###
115          if self.migration:
116             userInterface = session['userInterface']
117             self.logger.info("userInterface    " + userInterface)
118          else:
119          ###end migration code###
120             userInterface = None
121 
122          if self.useIPSEC:
123             ###migration code->added parameters: userInterface and
                    self.migration###
124             newTVD = UserTVD(session['token'], vlanID, self.
                networkManager, self.computeManager, self.config,
125                              self.logger, userInterface, self.
                                 migration, self.mobility)
126 
127          else:
128             ###migration code->added parameters: userInterface and
                    self.migration###
129             newTVD = UserTVDIPSECLess(session['token'], vlanID,
                self.networkManager, self.computeManager, self.
                config,
130                                       self.logger, userInterface,
                                          self.migration)
131 
132 
133          self.userTVDs[session['token']] = newTVD
134 
135          # Generates the PSC for the user
136          ###start migration code###
137          if self.migration:
138             mac = session['psc']['interfaces'][0]['mac']
139          else:
140          ###end migration code###
141             mac = self.networkManager.generateMACaddress()
142 
143          ###start migration code###
144          if self.migration:
145             newPSC = session['psc']
146          else:
147          ###end migration code###
148             newPSC = self.definePSC(vlanID, mac)
149 
150          ###start migration code###
151          if self.migration:
```

```
152             pscAddr = session['pscAddr']
153         else:
154         ###end migration code###
155             pscAddr = self.networkManager.configureNewIPOnDHCP(mac
                    , session['token'])
156 
157         self.logger.info("PSC address for user " + session['token
                '] + " " + pscAddr)
158         self.TokenIP[pscAddr] = session['token']
159         ###start migration code###
160         if self.migration:
161             pscName = session['pscName']
162         else:
163         ###end migration code###
164             pscName = self.computeManager.instantiateNF('psc',
                    newPSC)
165 
166         ##newPSC['name'] = pscName
167         ##newTVD.setPSC(newPSC, pscAddr)
168         ##self.logger.info("PSC created")
169 
170         newPSC['name'] = pscName
171         newTVD.setPSC(newPSC, pscAddr)
172         newTVD.generatePSCflows()
173 
174         self.timestamps["tvdm2_finish_instantiation_PSC"] =
                datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
175         self.logger.info("PSC created")
176 
177         newTVD.addNewIP(str(session['IP']))
178 
179         ###start migration code###
180         if self.mobility:
181             try:
182                 headers = {'Content-Type': 'application/json', '
                        Accept': 'application/json'}
183 
184                 if not 'ssid' in session:
185                     info = self.local_info
186                     self.logger.info("NAT TAKE LOCAL_INFO: %s" %
                            str(self.local_info))
187                 else:
188                     info = self.config.migration_ned_info(session
                            ['ssid'])
189 
190                     self.logger.info("NAT TAKE SSID %s INFO: %s" %
                             (ssid,str(info)))
191 
192                 if type(info) is str:
193                         info = ast.literal_eval(info)
194                 nat = info['nat']
195                 url = "http://%s:%s/switch" % (str(nat['server
                        '][0]),str(nat['server'][1]))
196                 payload = {'route': nat['route']}
197                 self.logger.info("NAT Request URL: %s" % url)
198 
```

```
199                 self.logger.info("NAT Request Payload: %s" % str(
                        payload))
200                 r = requests.post(url, json=payload, headers=
                        headers )
201                 self.logger.info("NAT request Response: %s" % str(
                        r.status_code))
202             except Exception as e:
203                 self.logger.error("NAT request error\n%s" % (str(e
                        )))
204
205         if self.migration:
206             psaIPaddresses = session['psaIPaddresses'].items()
207             for psa in  psaIPaddresses:
208                 psaID = psa[0]
209                 ip = psa[1]
210                 newTVD.associateIPPSA(psaID, ip)
211
212
213         if self.migration:
214             self.instantiatePSA(session)
215             self.timestamps["tvdm2_finish_instantiation_PSAs"] =
                    datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%f")
216             self.logger.info(str(self.timestamps) )
217             ###end migration code###
218
219
220
221     def definePSC(self, vlanID, mac):
222         '''
223         Define the template for the PSC of the TVD
224         '''
225
226         properties = {}
227         properties['memory'] = "512"
228         properties['vcpu'] = "1"
229         properties['interfaces'] = []
230
231         interface = {}
232         interface['mac'] = mac
233         interface['bridge'] = "brCtl"
234         interface['name'] = self.networkManager.generateVMPort()
235         properties['interfaces'].append(interface)
236
237         interface = {}
238         interface['vlan'] = vlanID
239         interface['bridge'] = "brCtl"
240         interface['name'] = self.networkManager.generateVMPort()
241         properties['interfaces'].append(interface)
242
243         interface = {}
244         interface['bridge'] = "brData"
245         interface['name'] = self.networkManager.generateVMPort()
246         properties['interfaces'].append(interface)
247
248         return properties
249
```

```
250     def deleteUser(self, session, migration=False):
251         '''
252         This function is used for the user log out. If there are
                multiple devices connected it will be deleted only the
253         flows for the specifc device that did the log out.
254         If the last device for that specific user is disconnected
                the graph will be destroyed
255         '''
256         if session['IP'] not in self.IPandUser.keys():
257             self.logger.info("Machine not logged in")
258             return
259         token = self.IPandUser[session['IP']]
260 
261         if self.mobility:
262             self.mobility_iprules(session, "delete", token)
263 
264         del self.IPandUser[session['IP']]
265         userTVD = self.userTVDs[token]
266 
267 
268         if userTVD.deleteTVD(session['IP'], migration=migration):
269             del self.TokenIP[userTVD.pscAddr]
270             del self.userTVDs[token]
271 
272     def instantiatePSA(self, session):
273         '''
274         Instantiate the PSA of the TVD
275         '''
276         global err, err
277         PSAList = session['PSAs']
278         self.logger.info("------> PSAList %s \n" %(json.dumps(
                PSAList)))
279         oup = "egress flow: "
280         if 'egress_flow' in session:
281             for psa in session['egress_flow']:
282                 oup = oup + str(psa) + ", "
283         oup = oup + "\n ingress flow: "
284         if 'ingress_flow' in session:
285             for psa in session['ingress_flow']:
286                 oup = oup + str(psa) + ", "
287         self.logger.info("\n%s\n" %(str(oup)))
288 
289         userTVD = self.userTVDs[session['token']]
290         userTVD.instantiatePSA(PSAList)
291 
292         self.logger.info("\n\n--> of FLOWS\n %s \n" %(json.dumps(
                userTVD.generatedFlows)))
293         self.logger.info("\n\n--> ofPorts\n %s \n" %(json.dumps(
                userTVD.ofPorts)))
294 
295         #######start migration code########
296         if self.mobility:
297             self.mobility_iprules(session, "add", None)
298         #######stop migration code#########
299 
300 
```

```
301     def signal_term_handler(self):
302         '''
303         Used on SIGTERM signal to destroy all the instantiated TVD
304         '''
305         if not self.sigTerm:
306             self.sigTerm = True
307             for userTVD in self.userTVDs.values():
308                 userTVD.deleteAllTVD()
309             return True
310         return False
311 
312     def get_default_gw(self, netns="default"):
313         '''
314         function to return the default gw ip
315         '''
316         strs = subprocess.check_output(shlex.split('ip netns exec
                '+ str(netns)+' ip r l'))
317         gateway = strs.split('default via')[-1].split()[0]
318         return gateway
319 
320     def mobility_iprules(self,session, adddel, token):
321 
322         if (adddel == "add"):
323             userTVD = self.userTVDs[session['token']]
324         else:
325             userTVD = self.userTVDs[token]
326 
327         psaIPaddresses = userTVD.psaIPaddresses.items()
328         ssid = self.config.DEFAULT_SSID
329         info = self.config.migration_ned_info(ssid)
330         if type(info) is str:
331             info = ast.literal_eval(info)
332         nat = info['nat']
333         try:
334             if 'default_gw' in nat:
335                 self.gw_ip = str(nat['default_gw'])
336             else:
337                 self.gw_ip = str(nat['server'][0])
338         except Exception as err:
339             self.logger.info("----> error getting the default gw "
                    + str(err))
340             self.gw_ip = str(nat['server'][0])
341 
342         for psa in psaIPaddresses:
343             psaID = psa[0]
344             ip_psa = psa[1]
345             self.logger.info("psaID " + psaID + " ip " + ip_psa)
346             if (adddel == "add"):
347                 self.psa_ip_route_table += 1
348                 if session['token'] not in userTVD.iprules or type
                    (userTVD.iprules[session['token']]) is not
                    dict:
349                     userTVD.iprules[session['token']] = {}
350                 if not ip_psa in userTVD.iprules[session['token
                    ']]:
351                     userTVD.iprules[session['token']][ip_psa] = []
```

```
352                 self.logger.info("[Mobility] ip_psa %s tables %s"
                        % (str(ip_psa), str(userTVD.iprules[session['
                        token']][ip_psa])))
353                 if not self.psa_ip_route_table in userTVD.iprules[
                        session['token']][ip_psa]:
354                     self.logger.info("Cleaning ip route table %s
                            ip_psa %s" % (str(self.psa_ip_route_table)
                            ,str(ip_psa)))
355                     commands.cleanRouteTable(table=str(self.
                            psa_ip_route_table), netNs="default")
356                 userTVD.iprules[session['token']][ip_psa].append(
                        self.psa_ip_route_table)
357                 ##add ip rule from
358                 commands.addIPrule(table=self.psa_ip_route_table,
                        addressFrom=ip_psa, pref=self.
                        psa_ip_route_table, netns="default")
359                 ##add ip rule to
360                 commands.addIPrule(table=self.psa_ip_route_table,
                        addressTo=ip_psa, pref=self.psa_ip_route_table
                        , netns="default")
361                 ##add ip routes to the table
362                 result = commands.addRoute(table=self.
                        psa_ip_route_table, addr=str(self.gw_ip),
                        default=True, netNs="default")
363                 if result is not None: self.logger.warning("\n\n[
                        mobility ip route] error add default route via
                         %s table %s" % (str(self.gw_ip), str(self.
                        psa_ip_route_table)))
364                 result = commands.addRoute(table=self.
                        psa_ip_route_table, addr=str(ip_psa), via=str(
                        self.ip_ext), netNs="default")
365                 if result is not None: self.logger.warning("\n\n[
                        mobility ip route] error add route to %s via %
                        s" % (str(self.ip_ext), str(ip_psa)))
366             else:
367                 tables = []
368                 if ip_psa in userTVD.iprules[token]:
369                     tables = userTVD.iprules[token][ip_psa]
370                     self.logger.info("Cleaning ip route table and
                            rules ip_psa %s tables %s" % (str(ip_psa),
                             str(tables)))
371                 for table in tables:
372                     commands.delIPrule(table=table, addressFrom=
                            ip_psa, netns="default")
373                     commands.delIPrule(table=table, addressTo=
                            ip_psa, netns="default")
374                     commands.cleanRouteTable(table=str(table),
                            netNs="default")
```

Fragment de codi B.4: userTVD.py

```
1 import commands
2 import json
3
4 from manifestManager import ManifestManager
5
```

```
6
7  class UserTVD(object):
8      '''
9      User TVD class in case of IPSEC NED configuration
10     '''
11
12     def __init__(self, userName, vlanID, networkManager,
           computeManager, config, logger, userInterface,
13                 migration=False, mobility=False):
14
15         self.logger = logger
16         self.networkManager = networkManager
17         ###start migration code###
18         if migration:
19             self.userInterface = userInterface
20             commands.createNewPort('brData', self.userInterface)
21             self.logger.info("entra migracio")
22         else:
23             ###end migration code###
24             self.userInterface = self.networkManager.generatePort
                   ('brData')
25             self.logger.info("entra intanciacio")
26
27         self.interfaceIP, result = self.networkManager.
               generateRoutingTable(self.userInterface)
28         self.logger.debug("\n\n->>> [RoutingTable] Interface %s,
               IP %s result: \n %s" % (str(self.userInterface), str(
               self.interfaceIP), str(result)))
29         self.userName = userName
30         self.userIP = []
31         self.vlanID = vlanID
32         self.pscAddr = None
33         self.psc = None
34         self.generatedFlows = []
35         self.computeManager = computeManager
36         self.config = config
37         self.psaList = []
38         self.psaIPaddresses = {}
39         self.manifestManager = ManifestManager(config)
40         self.migration = migration
41         self.iprules = {}
42         self.psaID_first = None
43         self.psaID_last = None
44         self.mobility = mobility
45         self.ofPorts = {}
46
47     def addNewIP(self, newIP):
48         '''
49             Add a new machine on the TVD with the given IP
50         '''
51         if newIP not in self.userIP:
52             self.userIP.append(newIP)
53             commands.addIPrule(table=self.userInterface,
                   addressFrom=newIP + "/32", pref=2, netns="default
                   ")
54             commands.addIPrule(table=self.userInterface, addressTo
```

```
                =newIP + "/32", pref=2, netns="default")
55
56      def delUserIP(self, newIP):
57          '''
58          Remove the flow for the given IP
59          '''
60          self.userIP.remove(newIP)
61          commands.delIPrule(table=self.userInterface, addressFrom=
                newIP + "/32", pref=2, netns="default")
62          commands.delIPrule(table=self.userInterface, addressTo=
                newIP + "/32", pref=2, netns="default")
63
64      def setPSC(self, newPSC, pscAddr):
65          '''
66      Configure the PSC of the TVD
67      '''
68          self.psc = newPSC
69          self.pscAddr = pscAddr
70          self.logger.info("User " + self.userName + " PSC addr: " +
                pscAddr)
71
72      def generatePSCflows(self):
73          '''
74          Generete the flow for the PSC
75          '''
76          flow = {}
77          bridgeName = 'brData'
78          flow['priority'] = "10"
79          match = {}
80          match['in_port'] = self.userInterface
81          self.ofPorts[self.userInterface] = commands.findPort(
                bridgeName, self.userInterface)
82          match['dl_type'] = "0x0806"
83          match['nw_dst'] = self.config.PSC_DP_IF_IP
84          flow['match'] = match
85          action = {}
86          action['output'] = self.psc['interfaces'][2]['name']
87          self.ofPorts[self.psc['interfaces'][2]['name']] = commands
                .findPort(bridgeName, self.psc['interfaces'][2]['name
                '])
88          flow['action'] = action
89          self.networkManager.generateFlow('brData', flow)
90          self.generatedFlows.append(flow)
91
92          flow = {}
93          flow['priority'] = "10"
94          match = {}
95          match['in_port'] = self.userInterface
96          match['dl_type'] = "0x0800"
97          match['nw_dst'] = self.config.PSC_DP_IF_IP
98          flow['match'] = match
99          action = {}
100         action['output'] = self.psc['interfaces'][2]['name']
101         flow['action'] = action
102         self.networkManager.generateFlow('brData', flow)
103         self.generatedFlows.append(flow)
```

```
104
105         flow = {}
106         match = {}
107         match['in_port'] = self.psc['interfaces'][2]['name']
108         flow['match'] = match
109         action = {}
110         action['output'] = self.userInterface
111         flow['action'] = action
112         self.networkManager.generateFlow('brData', flow)
113         self.generatedFlows.append(flow)
114
115     def deleteAllTVD(self):
116         '''
117         Delete the TVD
118         '''
119         self.logger.info("Deleting " + self.userName + " TVD")
120         for ip in self.userIP:
121             self.deleteTVD(ip)
122
123     def deleteTVD(self, IPaddr, migration=False):
124         '''
125         Remove an IP from the TVD in case of the last IP the TVD
                will be deleted
126         '''
127         self.logger.info("Removing IP " + IPaddr)
128         self.delUserIP(IPaddr)
129         if len(self.userIP) > 0:
130             return False
131         self.logger.info("Deleting flows: %s" % (str(self.
                generatedFlows)))
132         for flow in self.generatedFlows:
133             if migration is False:
134                 self.networkManager.deleteFlow('brData', flow)
135             else:
136                 self.networkManager.deleteFlow_migration('brData',
                        flow, ofPorts=self.ofPorts)
137
138
139         self.logger.info("Deleting the user Interface")
140         self.networkManager.deletePort('brData', self.
                userInterface)
141
142         ##self.logger.info("Deleting PSC %s\n" % (json.dumps(self.
                psc, indent=4, sort_keys=True)))
143         if self.psc is not None:
144             self.computeManager.deleteNF(self.psc['name'])
145
146         self.logger.info("Deleting PSA")
147         for psa in self.psaList:
148             self.computeManager.deleteNF(psa['name'])
149
150         logLine = "PSA:"
151         for ipAddr in self.psaIPaddresses.values():
152             logLine = logLine + " " + ipAddr
153         logLine = logLine + ", PSC: " + self.pscAddr + " removed"
154         self.logger.info(logLine)
```

```
155         return True
156 
157     def deleteFlows(self,IPaddr):
158 
159         for flow in self.generatedFlows:
160             self.networkManager.deleteFlow_migration('brData',
                    flow)
161             self.generatedFlows.remove(flow)
162 
163     def associateIPPSA(self, psaID, ip=None):
164         '''
165         Associate an IP on a PSA
166         '''
167 
168         ###start migration code####
169         if self.migration:
170             ipAddr = ip
171             self.psaIPaddresses[psaID] = ip
172         else:
173         ###end migration code###
174             ipAddr = self.networkManager.getPSAnewAddress()
175             self.psaIPaddresses[psaID] = ipAddr
176         self.logger.info("User " + self.userName + " PSA " + psaID
                + " addr: " + ipAddr)
177 
178 
179     def definePSA(self, psaID):
180         '''
181         Define a new PSA for the TVD
182         '''
183         manifest = self.manifestManager.getManifest(psaID)
184         properties = {}
185         properties['memory'] = manifest['memory']
186         properties['vcpu'] = manifest['vcpu']
187         properties['interfaces'] = []
188 
189         interface = {}
190         interface['bridge'] = "brData"
191         interface['name'] = self.networkManager.generateVMPort()
192         properties['interfaces'].append(interface)
193 
194         interface = {}
195         interface['bridge'] = "brData"
196         interface['name'] = self.networkManager.generateVMPort()
197         properties['interfaces'].append(interface)
198 
199         interface = {}
200         interface['vlan'] = self.vlanID
201         interface['bridge'] = "brCtl"
202         interface['name'] = self.networkManager.generateVMPort()
203         properties['interfaces'].append(interface)
204 
205         return properties
206 
207     def instantiatePSA(self, PSAList):
208         '''
```

```
209         Instantiate the PSAs for the TVD
210         '''
211         for psa in PSAList:
212             ###start migration code####
213             if self.migration:
214                 psaProperties = psa
215             else:
216                 ###end migration code###
217                 psaProperties = self.definePSA(psa['id'])
218             ###start migration code###
219             if self.migration:
220                 psaName = psa['name']
221                 for interface in psa['interfaces']:
222                     self.logger.info("psa interface " + interface
                            ['bridge'] + " " + interface['name'])
223                     #commands.createNewPort(interface['bridge'],
                            interface['name'])
224             else:
225                 ###end migration code###
226                 psaName = self.computeManager.instantiateNF(psa['
                        id'], psaProperties)
227             psaProperties['name'] = psaName
228             for interface in psaProperties['interfaces']:
229                 self.logger.info("psa interface " + interface['
                        bridge'] + " " + interface['name'])
230                 self.logger.info("lastinterface " + self.
                        userInterface)
231             self.psaList.append(psaProperties)
232
233         self.logger.info("\n\n [userTVD] PSAList:\n%s" % (json.
                dumps(self.psaList, indent=4, sort_keys=True)))
234         lastInterface = self.userInterface
235         for psa in self.psaList:
236             flow = {}
237             bridgeName = 'brData'
238             match = {}
239             match['in_port'] = lastInterface
240             self.ofPorts[lastInterface] = commands.findPort(
                    bridgeName, lastInterface)
241             flow['match'] = match
242             action = {}
243             action['output'] = psa['interfaces'][0]['name']
244             self.ofPorts[psa['interfaces'][0]['name']] = commands.
                    findPort(bridgeName, psa['interfaces'][0]['name'])
245             flow['action'] = action
246             self.networkManager.generateFlow('brData', flow)
247             self.generatedFlows.append(flow)
248             flow = {}
249             match = {}
250             match['in_port'] = psa['interfaces'][0]['name']
251             flow['match'] = match
252             action = {}
253             action['output'] = lastInterface
254             flow['action'] = action
255             self.logger.info("flows2 " + str(flow))
256             self.networkManager.generateFlow('brData', flow)
```

```
257                 self.generatedFlows.append(flow)
258                 lastInterface = psa['interfaces'][1]['name']
259             flow = {}
260             bridgeName = 'brData'
261             match = {}
262             match['in_port'] = lastInterface
263             self.ofPorts[lastInterface] = commands.findPort(bridgeName
                    , lastInterface)
264             flow['match'] = match
265             action = {}
266             action['output'] = self.config.EXIT_INTERFACE
267             self.ofPorts[self.config.EXIT_INTERFACE] = commands.
                    findPort(bridgeName, self.config.EXIT_INTERFACE)
268             flow['action'] = action
269             self.logger.info("flows3 " + str(flow))
270             self.networkManager.generateFlow('brData', flow)
271             self.generatedFlows.append(flow)
272 
273             flow = {}
274             flow['priority'] = "10"
275             bridgeName = 'brData'
276             match = {}
277             match['in_port'] = self.config.EXIT_INTERFACE
278             match['dl_type'] = "0x0806"
279             match['nw_dst'] = self.interfaceIP
280             flow['match'] = match
281             action = {}
282             action['output'] = lastInterface
283             self.ofPorts[lastInterface] = commands.findPort(bridgeName
                    , lastInterface)
284             flow['action'] = action
285             self.logger.info("flows4 " + str(flow))
286             self.networkManager.generateFlow('brData', flow)
287             self.generatedFlows.append(flow)
288 
289             flow = {}
290             flow['priority'] = "10"
291             bridgeName = 'brData'
292             match = {}
293             match['in_port'] = self.config.EXIT_INTERFACE
294             match['dl_type'] = "0x0800"
295             match['nw_dst'] = self.interfaceIP
296             flow['match'] = match
297             action = {}
298             action['output'] = lastInterface
299             self.ofPorts[lastInterface] = commands.findPort(bridgeName
                    , lastInterface)
300             flow['action'] = action
301             self.logger.info("flows5 " + str(flow))
302             self.networkManager.generateFlow('brData', flow)
303             self.generatedFlows.append(flow)
304 
305             for ipAddr in self.psaIPaddresses.values():
306                 flow = {}
307                 bridgeName = 'brData'
308                 flow['priority'] = "10"
```

```
309             match = {}
310             match['in_port'] = self.config.EXIT_INTERFACE
311             match['dl_type'] = "0x0806"
312             match['nw_dst'] = ipAddr
313             flow['match'] = match
314             action = {}
315             action['output'] = lastInterface
316             self.ofPorts[lastInterface] = commands.findPort(
                    bridgeName, lastInterface)
317             flow['action'] = action
318             self.logger.info("flows6 " + str(flow))
319             self.networkManager.generateFlow('brData', flow)
320             self.generatedFlows.append(flow)
321 
322             flow = {}
323             flow['priority'] = "10"
324             match = {}
325             match['in_port'] = self.config.EXIT_INTERFACE
326             match['dl_type'] = "0x0800"
327             match['nw_dst'] = ipAddr
328             flow['match'] = match
329             action = {}
330             action['output'] = lastInterface
331             flow['action'] = action
332             self.logger.info("flows7 " + str(flow))
333             self.networkManager.generateFlow('brData', flow)
334             self.generatedFlows.append(flow)
335 
336         for ipAddr in self.userIP:
337             flow = {}
338             flow['priority'] = "10"
339             match = {}
340             match['in_port'] = self.config.EXIT_INTERFACE
341             match['dl_type'] = "0x0806"
342             match['nw_dst'] = ipAddr
343             flow['match'] = match
344             action = {}
345             action['output'] = lastInterface
346             flow['action'] = action
347             self.logger.info("flows8 " + str(flow))
348             self.networkManager.generateFlow('brData', flow)
349             self.generatedFlows.append(flow)
350 
351             flow = {}
352             flow['priority'] = "10"
353             match = {}
354             match['in_port'] = self.config.EXIT_INTERFACE
355             match['dl_type'] = "0x0800"
356             match['nw_dst'] = ipAddr
357             flow['match'] = match
358             action = {}
359             action['output'] = lastInterface
360             flow['action'] = action
361             self.logger.info("flows9 " + str(flow))
362             self.networkManager.generateFlow('brData', flow)
363             self.generatedFlows.append(flow)
```

B.2 Codi amb les noves funcionalitats de la mobilitat

Fragment de codi B.5: TVDMigration.py

```
import ast
import datetime
import json
import libvirt
import requests
import socket
import subprocess
import threading
import time
from threading import Event
from threading import Thread
import paramiko

from libvirtManager import ComputeManager
class TVDMigration(object):
    '''
       Class used to migrate user's TVD and VMs
    '''

    def __init__(self, instantiator, configure, logger):
        self.user = None
        self.userTVDs = instantiator.userTVDs  # All the generated
            TVD
        self.remote_conn_string = ""
        self.local_conn_string = ""
        self.conn = ""
        self.remote_conn = ""

        self.default_ssid = configure.DEFAULT_SSID
        self.local_info = configure.migration_ned_info(self.
           default_ssid)
        if type(self.local_info) is str:
            self.local_info = ast.literal_eval(self.local_info)
        self.local_host = self.local_info['ned_ip']
        self.local_port = int(self.local_info['ned_port'])

        self.config = configure
        self.computeManager = ComputeManager(configure)
        self.__last_bytes = -1
        self.__last_time = datetime.time()
        self.logger = logger
        self.PSAList = {}
        self.TVD = {}
        self.namePSC = None
        self.instantiator = instantiator
        self.TokenIP = instantiator.TokenIP
        self.ssh = None
        self.PSC = None
        self.disk_base = None
        self.local_conn_string = "qemu:///system"
        self.conn = libvirt.open(self.local_conn_string)
        self.local_dom = None
```

```
51
52      def init_migration(self, e, he, session):
53          '''
54          Inicialitzation class and call migration functions
55          '''
56
57          self.logger.info("token " + str(session['token']))
58
59          self.instantiator.psas_tt = {}
60          self.remote_info = self.config.migration_ned_info(session
                ['ssid'])
61          if type(self.remote_info) is str:
62              self.remote_info = ast.literal_eval(self.remote_info)
63          self.remote_host = self.remote_info['ned_ip']
64          self.remote_port = int(self.remote_info['ned_port'])
65          self.logger.info("remote host %s:%s" % (self.remote_host,
66           self.remote_port))
67          self.PSAList = self.userTVDs[session['token']].psaList
68          self.user = self.userTVDs[session['token']].userName
69          self.namePSC = self.userTVDs[session['token']].psc['name']
70          self.PSC = self.instantiator.userTVDs[session['token']].
                psc
71          try:
72              self.ssh = self.createSSHClient(self.remote_host, self
                    .remote_port,
73               'root', 'secured')
74          except:
75              self.logger.info("Error to generate ssh")
76          i = Event()
77          self.TsendTVD = Thread(target=self.sendTVD, kwargs={"
                session": session})
78          Migration = Thread(target=self.migration, args=(i, he, ))
79          Migration.start()
80          i.wait()
81          e.set()
82          Migration.join()
83          self.logger.info("Migration FINISHED. Deleting remaining
                user data... ")
84          self.sendTimestamps(session)
85          self.logger.info("calling instantiator.deleteUser with
                session: \n%s" %
86          (json.dumps(session, indent=4, sort_keys=True)))
87          self.instantiator.deleteUser(session, migration=True)
88          self.logger.info("\n\nUSER DELETED... SHOULD BE CLEAN... \
                n")
89
90
91      def sendTimestamps(self, session):
92          obj = {"action":"timestamps"}
93
94          #4#
95          if self.instantiator.tvdm_receives_migration_request:
96              self.instantiator.timestamps["
                    tvdm_receives_migration_request"] =
97              self.instantiator.tvdm_receives_migration_request
98          #5#
```

```
99          if self.instantiator.psas_tt:
100             self.instantiator.timestamps["psas_tt"] =
101             self.instantiator.psas_tt
102         #6#
103         if self.instantiator.tvdm_ini_psc:
104             self.instantiator.timestamps["tvdm_ini_psc"] =
105             self.instantiator.tvdm_ini_psc
106         #7#
107         if self.instantiator.tvdm_sends_TVD:
108             self.instantiator.timestamps["tvdm_sends_TVD"] =
109             self.instantiator.tvdm_sends_TVD
110         #8#
111         if self.instantiator.tvdm_fin_psc:
112             self.instantiator.timestamps["tvdm_fin_psc"] =
113             self.instantiator.tvdm_fin_psc
114         #9#
115         if self.instantiator.tvdm_finishes_migration:
116             self.instantiator.timestamps["tvdm_finishes_migration
                    "] =
117             self.instantiator.tvdm_finishes_migration
118         #10#
119         if self.instantiator.tvdm_sends_migration_finished:
120             self.instantiator.timestamps["
                    tvdm_sends_migration_finished"] = self.
                    instantiator.tvdm_sends_migration_finished
121         #11---------------------#
122         '''
123         if self.instantiator.tvdm2_recieves_TVD:
124             self.instantiator.timestamps["tvdm2_recieves_TVD"] =
                    self.instantiator.tvdm2_recieves_TVD
125         #12#
126         if self.instantiator.tvdm2_finish_instantiation_PSAs:
127             self.instantiator.timestamps["
                    tvdm2_finish_instantiation_PSAs"] = self.
                    instantiator.tvdm2_finish_instantiation_PSAs
128         #13#
129         if self.instantiator.tvdm2_finish_instantiation_PSC:
130             self.instantiator.timestamps["
                    tvdm2_finish_instantiation_PSC"] = self.
                    instantiator.tvdm2_finish_instantiation_PSC
131         '''
132         obj["timestamps"] = self.instantiator.timestamps
133         try:
134             timestamps = json.dumps(obj)
135             self.logger.info("TIMESTAMPS: " + str(timestamps))
136         except:
137             self.logger.info("Error to created TVD json")
138
139         try:
140             cmd = "ip netns exec orchNet curl -X PUT --header
                    Accept: application/json --header Content-Type:
                    application/json -d '" + timestamps + "' http
                    ://192.168.1.1:8080/migration"
141             (results, errors) = self.execute_command(self.ssh, cmd
                    , False)
142
```

```
143         except:
144             self.logger.info(errors)
145 
146     def sendTVD(self, session):
147         '''
148         Obtain user's TVD and migrate this
149         '''
150 
151         self.TVD['token'] = self.instantiator.userTVDs[session['
                token']].userName
152         self.TVD['IP'] = session['IP']
153         self.TVD['userInterface'] =
154          self.instantiator.userTVDs[session['token']].
                userInterface
155         self.instantiator.logger.info("userIsessionnterface " +
156          str(self.TVD['userInterface']))
157         self.TVD['vlanID'] = self.instantiator.userTVDs[session['
                token']].vlanID
158         self.TVD['pscAddr'] =
159         self.instantiator.userTVDs[session['token']].pscAddr
160         self.TVD['psc'] = self.instantiator.userTVDs[session['
                token']].psc
161         self.TVD['pscName'] =
162          self.instantiator.userTVDs[session['token']].psc['name']
163         self.TVD['generatedFlows'] =
164          self.instantiator.userTVDs[session['token']].
                generatedFlows
165         self.TVD['PSAs'] = self.instantiator.userTVDs[session['
                token']].psaList
166         self.TVD['psaIPaddresses'] =
167          self.instantiator.userTVDs[session['token']].
                psaIPaddresses
168         self.TVD['migration'] = "True"
169         self.TVD['action'] = "TVD"
170 
171         try:
172             TVD = json.dumps(self.TVD)
173             self.logger.info("TVD: " + str(TVD))
174         except:
175             self.logger.info("Error to created TVD json")
176 
177         try:
178             cmd = "ip netns exec orchNet curl -X PUT --header \
179             Accept: application/json --header Content-Type:
                    application/json -d\
180              '" + TVD + "' http://192.168.1.1:8080/migration"
181             (results, errors) = self.execute_command(self.ssh, cmd
                    , False)
182             self.instantiator.tvdm_sends_TVD =
183              datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M:%
                    S.%f")
184 
185         except:
186             self.logger.info(errors)
187 
188     def migration(self, i, he):
```

```
189         '''
190         Prepare PSC and PSAs migration
191         '''
192
193         '''
194         PSAs migration
195         '''
196         tpsas = []
197         for psa in self.PSAList:
198             self.original = self.obtain_disk_base(psa['disk']['
                location'])
199             self.generate_remote_disk(self.original, psa['disk']['
                type'], psa['disk']['location'], psa['name'])
200             self.logger.info("going to migrate "+ psa['name'])
201             tpsa = TMigration(vm=psa['name'], sleep=0,
202                               local_host=self.local_host,
203                               remote_host=self.remote_host,
204                               remote_port=self.remote_port,
205                               logger=self.logger)
206             tpsa.start()
207             ini = datetime.datetime.utcnow().strftime("%Y-%m-%d %H
                :%M:%S.%f")
208             obj = {"ini": ini}
209             self.instantiator.psas_tt[psa['name']] = obj
210
211             tpsas.append(tpsa)
212
213         while 1:
214             avgr = 0
215             for tpsa in tpsas:
216                 remaining = tpsa.vm_status()
217                 if remaining[0] > 0:
218                     avgr += remaining[0]
219             if len(tpsas) > 0:
220                 if avgr / len(tpsas) >= 80:
221                     i.set()
222                     self.instantiator.tvdm_finishes_migration =
223                     datetime.datetime.utcnow().strftime("%Y-%m-%d
                        %H:%M:%S.%f")
224                     self.logger.info("---> PSAs migrated [event: %
                        s] \
225                     " % (str(i.isSet())))
226                     break
227
228         self.logger.info("---> waiting for handover ack to the PSC
                [event: %s \
229         ]" % (str(i.isSet())))
230         he.wait()  # wait for the handover answer to the PSC
231
232         '''
233         PSC migration
234         '''
235         self.original_psc = (self.obtain_disk_base(self.PSC['disk
            ']['location']))
236         self.generate_remote_disk(self.original_psc, self.PSC['
            disk']['type'],
```

```
                                    self.PSC['disk']['location'],
                                       self.namePSC)
        tMigration = TMigration(vm=self.namePSC,
                                sleep=2,
                                local_host=self.local_host,
                                remote_host=self.remote_host,
                                remote_port=self.remote_port,
                                logger=self.logger)
        tMigration.start()
        print ("tvdm_ini_psc puestooooooooooooo")
        self.instantiator.tvdm_ini_psc =
          datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M:%S.%
             f")

        '''
        TVD migration
        '''
        try:
            nat = self.remote_info['nat']
            headers = {'Content-Type': 'application/json',
                       'Accept': 'application/json'}
            payload = {'route': nat['route']}
            url = "http://%s:%s/switch" % (str(nat['server'][0]),
                                           str(nat['server'][1]))
            self.logger.info("NAT Request URL: %s" % url)
            r = requests.post(url, json=payload, headers=headers)
            self.logger.info("NAT request Response: %s" % str(r.
               status_code))
        except Exception as e:
            self.logger.error("NAT request error\n%s" % (str(e)))

        time.sleep(1)

        self.TsendTVD.start()  # pass TVD, after init PSC
           migration

        while 1:
            if tMigration.executed:
                break
        remaining = tMigration.vm_status()
        while remaining:
            if not remaining:
                self.logger.info("No migration in progress")
            else:
                self.logger.info("remaining to migrate %s, psa: %s
                    " %
                                 (str(remaining), str(tMigration.
                                    vm)))
            time.sleep(1)
            remaining = tMigration.vm_status()
            if remaining[0] >= 80:
                # subprocess.check_call()
                break

        for tpsa in tpsas:
            tpsa.join()
```

```
287                 self.instantiator.psas_tt[tpsa.vm]["fin"] =
288                       datetime.datetime.utcnow().strftime("%Y-%m-%d %H
                          :%M:%S.%f")
289
290             self.TsendTVD.join()
291             tMigration.join()
292             self.instantiator.tvdm_fin_psc =
293                    datetime.datetime.utcnow().strftime("%Y-%m-%d %H:%M
                       :%S.%f")
294
295             for psa in self.PSAList:
296                 self.undefined_vm(psa['name'])
297
298             self.undefined_vm(self.namePSC)
299             self.logger.info("PSC migrated")
300
301
302         def undefined_vm(self, vm):
303             self.local_dom = self.conn.lookupByName(vm)
304             self.local_dom.undefine()
305
306
307         def obtain_disk_base(self, path):
308             '''
309             obtain path VM disk base
310             '''
311             try:
312                 proc = subprocess.Popen("qemu-img info " + path,
313                                         stdout=subprocess.PIPE, shell
                                            =True)
314                 (out, err) = proc.communicate()
315                 returnedValue = str(out)
316                 start = 'file: '
317                 end = '\n'
318                 disk_base = ((returnedValue.split(start))[1].split(end
                    )[0])
319
320             except subprocess.CalledProcessError as e:
321                 self.logger.info("Calledprocerr:" + e)
322
323             return disk_base
324
325         def generate_remote_disk(self, original, disk_type, path, vm):
326             """
327             Generate remote disk
328             """
329             stderr = ""
330
331             try:
332                 cmd = "[  -f " + path + "/" + vm + " ] || qemu-img
                       create -b "
333                       + original + " -f " + disk_type + " " + path
334                 stdin, stdout, stderr = self.ssh.exec_command(cmd)
335                 stdin.close()
336             except Exception as e:
337                 self.logger.info("Error to create remote disk " + str(
```

```
                e) + " "
338                             + str(stderr))
339
340     def rename_remote_disk(self, path, vm):
341
342         stderr = ""
343
344         try:
345             cmd = "mv " + path + "/" + vm + " " + path + "/" + vm
                    + "backup"
346             stdin, stdout, stderr = self.ssh.exec_command(cmd)
347             stdin.close()
348
349         except Exception as e:
350             self.logger.info("Error to create remote disk "
351                             + str(e) + " " + str(stderr))
352
353
354     def createSSHClient(self, server, port, user, password):
355
356         client = paramiko.SSHClient()
357         client.load_system_host_keys()
358         client.set_missing_host_key_policy(paramiko.AutoAddPolicy
                ())
359         client.connect(server, port, user, password)
360         return client
361
362     def execute_command(self, ssh, cmd, sudo):
363         '''
364         Create remote conection
365         '''
366         try:
367             if sudo:
368                 cmd = "sudo -S -u qemu '' %s" % cmd
369             stdin, stdout, stderr = ssh.exec_command(cmd)
370             if sudo:
371                 stdin.write('xxx\n')
372             stdin.write('x\n')
373             stdin.flush()
374
375         except socket.timeout as serr:
376             self.logger.info("Error1: %s" % serr)
377
378         except socket.error as serr:
379             print "Error2: %s" % serr
380
381         except:
382             print "ANY1"
383         return (stdout.readlines(), stderr.readlines())
384
385
386 class TMigration(threading.Thread):
387     def __init__(self, vm, sleep, local_host, remote_host,
            remote_port,
388                  logger, group=None, target=None, name=None,args
                        =(),
```

```
389                     kwargs=None, verbose=None):
390         super(TMigration, self).__init__(group=group, target=
                target,
391                                          name=name, verbose=
                                             verbose)
392         self.vm = vm
393         self.sleep = sleep
394         self.local_host = local_host
395         self.remote_host = remote_host
396         self.remote_port = remote_port
397         self.logger = logger
398         self.local_conn_string = "qemu:///system"
399         __last_bytes = -1
400         __last_time = datetime.time()
401 
402         try:
403             self.remote_conn_string = "qemu+ssh://"
404                                       + self.remote_host + ":"
405                                       + str(self.remote_port) + "/
                                          system"
406         except:
407             self.logger.info("Error Connecting remote VM " + self.
                remote_host)
408 
409         try:
410             self.conn = libvirt.open(self.local_conn_string)
411         except:
412             self.logger.info("Error Connecting to VM hypervisor "
413                              + self.local_conn_string)
414         try:
415             self.remote_conn = libvirt.open(self.
                remote_conn_string)
416         except:
417             self.logger.info("Error connecting to remote
                hypervisor: "
418                              + self.remote_conn_string)
419 
420         if (self.sleep > 0): self.logger.info(time.sleep(self.
            sleep))
421 
422         self.logger.info(
423                         "Starting migration of VM domain '" + self.
                            vm + "' from "
424                         + self.local_host + " to " + self.
                            remote_host + "...")
425         try:
426             self.local_dom = self.conn.lookupByName(self.vm)
427         except:
428             self.logger.info("VM does not exist: " + self.vm)
429         self.executed = False
430         return
431 
432     def run(self):
433 
434         try:
435             self.logger.info("RUN DE TMIGRATION")
```

```
436             self.local_dom.migrate(self.remote_conn, libvirt.
                    VIR_MIGRATE_LIVE |
437                                    libvirt.VIR_MIGRATE_COMPRESSED,
                                           None,
438                                    None, 0)
439             self.logger.info("Migration of VM domain '" + self.vm
                    + "' from "
440                              + self.local_host + " to " + self.
                                     remote_host
441                              + "triggered OK!!")
442             self.executed = True
443         except Exception as e:
444             self.logger.info("Error to migrate " + self.vm)
445             self.logger.error(str(e))
446 
447         try:
448             remaining = self.vm_status()
449             while remaining[0] != 100:
450                 if not remaining:
451                     self.logger.info("No migration in progress")
452                 else:
453                     self.logger.info("tmigration remaining to
                            migrate %, psa: "
454                                      % (str(remaining), str(self.
                                             vm)))
455                 time.sleep(1)
456                 remaining = self.vm_status()
457         except Exception as e:
458             self.logger.info("Error GETTING STATUS " + self.vm)
459             self.logger.error(str(e))
460         return
461 
462     def vm_job_stats(self):
463         if self.local_dom.info()[0] != 5:
464             return self.local_dom.jobStats()
465         else:
466             return {}
467 
468     def __update_migration_status(self):
469         dictionary = self.vm_job_stats()
470         if dict:
471             if not 'data_remaining' in dictionary:  # No migration
                        in progress
472                 if self.executed:
473                     return [100]
474                 return [-1]
475             remaining_bytes = dictionary['data_remaining']
476             total_bytes = dictionary['data_total']
477             if total_bytes == 0:
478                 if self.executed:
479                     return [100]
480                 return [-1]
481             if self.__last_bytes < 0:
482                 self.__last_bytes = remaining_bytes
483                 self.__last_time = datetime.datetime.now();
484                 eta = datetime.timedelta()
```

```
485             elif remaining_bytes == 0:
486                 self.__last_bytes = 0
487                 self.__last_time = datetime.time();
488             else:
489                 byte_rate = self.__last_bytes - remaining_bytes
490                 time_now = datetime.datetime.now()
491                 time_between_queries = time_now - self.__last_time
492                 eta_microsec = self.timedelta2microseconds(
                        time_between_queries)
493                                 * int(remaining_bytes / byte_rate)
494                 eta = self.microseconds2timedelta(eta_microsec)
495                 self.__last_time = time_now
496                 self.__last_bytes = remaining_bytes
497             percent_done = remaining_bytes / total_bytes * 100
498             return [percent_done, eta.days, eta.seconds, eta.
                    microseconds]
499         else:
500             if self.executed:
501                 return [100]
502             return [-1]
503 
504     def vm_status(self):
505         try:
506             if not self.executed:
507                 return [-1]
508             return self.__update_migration_status()
509         except Exception as e:
510             self.logger.info("vm_status vm not found %s" % (str(e.
                    message)))
511             return [100]
512 
513     def timedelta2microseconds(self, time):
514         if not time:
515             return -1
516         return (((24 * 60 * 60 * time.days) + time.seconds)) * 10
                ** 6
517                 + time.microseconds
518 
519     def time2microseconds(self, time):
520         if not time:
521             return -1
522         return (((60 * time.hours) + time.minutes) * 60 + time.
                seconds)
523                 * 1000000 + time.microseconds
524 
525     def microseconds2timedelta(self, pass_microseconds):
526         given_microseconds = int(pass_microseconds)
527         microseconds = given_microseconds % (10 ** 6)
528         rest = int(given_microseconds / 10 ** 6)
529         seconds = int(rest % (60 * 60 * 24))
530         days = int(rest / (60 * 60 * 24))
531         return datetime.timedelta(days=days, seconds=seconds,
532                                   microseconds=microseconds)
533 
534     def microseconds2time(self, given_microseconds):
535         microseconds = given_microseconds % (10 ** 6)
```

```
536         rest = int(given_microseconds / 10 ** 6)
537         seconds = int(rest % 60)
538         rest = int(rest / 60)
539         minutes = int(rest % 60)
540         rest = int(rest / 60)
541         hours = int(rest % 60)
542         return datetime.time(hours=hours, minutes=minutes, seconds
                =seconds,
543                              microseconds=microseconds)
```

B.3 Bibliografia

Bibliografia

[1] SECURED: https://www.secured-fp7.eu

[2] SECURED, consortium: https://www.secured-fp7.eu/about/consortium

[3] Definiciió de OPEN-FLOW: https://www.opennetworking.org/sdn-resources/openflow

[4] Definició QEMU: https://ca.wikipedia.org/wiki/QEMU

[5] Descripció de Strongswan: https://www.strongswan.org

[6] Ley Organica de Protección de Datos: https://es.wikipedia.org/wiki/Ley_Org%C3%A1nica_de_Protecci%C3%B3n_de_Datos_de_Car%C3%A1cter_Personal_de_Espa%C3%B1a

[7] OpenDayLight: https://www.opendaylight.org/

[8] Openvswitch: http://openvswitch.org/

[9] API REST: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[10] Definició d'anti-phishing: https://en.wikipedia.org/wiki/Anti-phishing_software

[11] Informació de la llibreria libvirt: https://libvirt.org/migration.html

[12] Informació sobre JSON: https://ca.wikipedia.org/wiki/JSON

[13] Definició dels flags: http://whatis.techtarget.com/definition/flag

[14] Descripció de la llibreria Paramiko: http://www.paramiko.org

[15] Descripció de la llibreria libvirt https://libvirt.org/migration.html

[16] Infografia Sostenibilitat: https://libvirt.org/migration.html

[17] Documentació de Sharelatex per la relització del TFG: https://libvirt.org/migration.html

[18] Informació dels NUC: hhttps://en.wikipedia.org/wiki/Next_Unit_of_Computing

[19] Definició de la mobike: https://wiki.strongswan.org/projects/strongswan/wiki/MobIke